Docket No.: M4065.0573/P573-B
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of:
David C. Feldmeier et al.

Confirmation No.: 8591

Application No.: 10/645,604

Art Unit: 2186

Filed: August 22, 2003

Examiner: M.D. Anderson

For: PARTIALLY ORDERED CAMS USED IN TERNARY HIERARCHICAL ADDRESS SEARCHING/SORTING

APPEAL BRIEF

U.S. Patent and Trademark Office
Customer Window, Mail Stop Appeal Brief - Patents
Randolph Building
Alexandria, VA 22314

Dear Sir:

This is an Appeal Brief pursuant to 35 U.S.C. § 134 and 37 C.F.R. §§ 41.31 et seq. from the final rejection of claims 56-87 of the above-identified application. The Notice of Appeal was filed on June 21, 2005. The fee for submitting this Brief in accordance with 37 C.F.R. § 1.17(c) is attached. Any deficiency in the fees associated with this Brief should be charged to Deposit Account No. 04-1073 under Order No. M4065.0573/P573-B.

09/20/2005 SZEWDIE1 00000008 10645604
02 FC:1402 500.00 OP

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Micron Technology, Inc., a corporation organized under and pursuant to the laws of the United States, and the assignee of this application.

II. RELATED APPEALS AND INTERFERENCES

There are no other known appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Current Status of Claims

Canceled	1-55
Withdrawn	None
Pending	56-87
Allowed	None
Rejected	56-87

B. Claims on Appeal

The claims on appeal are claims 56-87.

IV. STATUS OF AMENDMENTS

A final Office Action was mailed March 3, 2005. Appellants filed a Request for Reconsideration on June 2, 2005. An Advisory Action was mailed June 10, 2005. Appellants filed a Notice of Appeal on June 20, 2005. No amendments have been submitted subsequent to the March 3, 2005 final Office Action.

A complete listing of the claims on appeal appears in Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claims 56, 62, 65, 71, 73, 77-81 are pending. Appellants respectfully note, however, that the present appeal is based entirely upon whether the references cited by the PTO are available as prior art. Appellants' appeal and related arguments are not based on the language of the pending claims. In any event, Pursuant to 37 C.F.R. §41.37(c), Appellants provide the following summary of the claimed subject matter on appeal.

Independent claim 56 is directed to a binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask. An example of a binary CAM according to claim 56 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The binary CAM 20 comprises a plurality of entries wherein: each entry comprises: (1) a first value comprising the logically ANDed communication system address and its associated mask; and (2) a second value comprising the logically ANDed complement of said communication system address and

its associated mask; and wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

Independent claim 62 is directed to a binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask. An example of a binary CAM according to claim 62 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The binary CAM 20 comprises a plurality of entries wherein: each entry comprises a two bit representation for each bit in said ternary hierarchical address; and wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

Independent claim 65 is directed to a method for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask. An example of a binary CAM 20 employing the method according to claim 65 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The method comprises the steps of: logically ANDing each communication system address and its associated mask to form a first value; logically ANDing the complement of each communication system address and its associated address mask to form a second value; and storing said first value and said second value in an entry in said binary CAM based on the number of contiguous ones in said address mask.

Independent claim 71 is directed to a method of storing for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask. An example of a binary CAM 20 employing the method according to claim 71 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The method comprises the steps of: (a) generating a CAM entry for each ternary hierarchical address by: (1) identifying that portion of each ternary hierarchical address that are don't cares; (2) representing each bit in said ternary hierarchical address using bits in said CAM wherein: (i) a 1 in said ternary hierarchical address is represented as 10 in said CAM; (ii) a 0 in said ternary hierarchical address is represented as 01 in said CAM; and (iii) a don't care in said ternary hierarchical address is represented as 00 in said CAM; (b) positioning each CAM entry in said CAM based on the number of contiguous ones in said associated mask.

Independent claim 73 is directed to a method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address entry comprising a communication system address and an associated communication system address mask, said entries of ternary hierarchical addresses being stored in said binary CAM as a first value and a second value, said first value comprising the communication system address logically ANDed with said address mask and said second value comprising the complement of said communication system address logically ANDed with said address mask, and wherein all of said entries are ordered in said binary CAM based on the number of contiguous ones in said mask. An example of a binary CAM 20 employing the method according to claim 73 is described in the specification at page 23,

lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The method comprises the steps of: (a) loading a first register with the communication system address to be searched along with the complement of the communication system address; (b) loading a second register with the communication system address to be searched along with the complement of the communication system address; (c) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM; (d) determining whether a bit match occurs between corresponding bits of said first register and each entry in said binary CAM as qualified by said corresponding bit of said second register; and (e) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

Independent claim 77 is directed to a method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address comprising a communication system address and an associated communication system address mask, each ternary hierarchical address entry being stored in said binary CAM whereby 10 is stored in said entry for every 1 in said ternary hierarchical address, 01 is stored in said entry for every 0 in said ternary hierarchical address, and 00 is stored in said entry for every don't care in said ternary hierarchical address, and wherein all of said entries are ordered in said binary CAM based on the number of contiguous ones in said mask. An example of a binary CAM 20 employing the method according to claim 77 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The method comprises the steps of: (a) loading a first register and second register with the communication system address to be searched by: (1) loading a 10 in said

first and second registers for every 1 in the ternary hierarchical address to be searched; (2) loading a 01 in said first and second registers for every 0 in the ternary hierarchical address to be searched; (3) loading a 00 in said first and second registers for every don't care in the ternary hierarchical address to be searched; (b) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM; (c) declaring a bit match between corresponding bits of said first register and each entry in said binary CAM: (1) if the corresponding bit in said second register is a 1; or (2) if the corresponding bit in said second register is a 0 and the corresponding bits of said first register and each entry in said binary CAM are identical; and (d) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

Independent claim 78 is directed to a method for maintaining a sorted CAM to enable longest matches in a single search cycle when hierarchical addresses are added to, or deleted from, the CAM in a communication system utilizing hierarchical addresses and associated address masks. An example of a binary CAM 20 employing the method according to claim 78 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The method comprises the steps of: (a) segmenting the CAM into blocks wherein each block corresponds to a single hierarchical mask and wherein said blocks are arranged in the CAM such that the lowest CAM addresses contain the highest hierarchical masks and the highest CAM addresses contain the lowest hierarchical masks; (b) storing hierarchical addresses according to said block having a corresponding hierarchical mask; and (c) tracking the first address and the next free address of each of said blocks and the size of each of said blocks.

Independent claim 79 is directed to a content addressable memory (CAM) of a communications system utilizing ternary hierarchical addressing and associated address masks. An example of a binary CAM according to claim 79 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The binary CAM 20 comprises: a plurality of address entries and associated address masks that are arranged in said CAM by mask number, with address entries having the highest mask number being located at address entry locations at the top of the CAM and address entries having the lowest mask number being located at address entry locations at the bottom of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

Independent claim 80 is directed to an apparatus for storing a plurality of address entries and associated address masks in a communication system utilizing ternary hierarchical addressing and associated address masks. An example of an apparatus according to claim 80 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The apparatus comprises: a binary CAM 20; a binary-encoded ternary converter 34A, 36 coupled to said binary CAM wherein said binary-encoded ternary converter converts each ternary value into a corresponding binary-encoded ternary value to form said address entries; and wherein said plurality of address entries and associated address masks are arranged in said binary CAM in a plurality of address entry groups, each address entry group having a respective mask number, with said address entry group having the highest mask number being located at the highest location of the CAM and said address entry group having the lowest mask number being located at the lowest location of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

Independent claim 81 is directed to an apparatus for storing a plurality of address entries and associated address masks in a communication system utilizing ternary hierarchical addressing and associated address masks. An example of an apparatus according to claim 80 is described in the specification at page 23, lines 4 to page 26, line 14 and illustrated at FIGS. 13, 15 and 16. The apparatus comprises: a binary CAM 20; a binary-encoded ternary converter 34A, 36 coupled to said binary CAM wherein said binary-encoded ternary converter converts each ternary value into a corresponding binary-encoded ternary value to form said address entries; and wherein said plurality of address entries and associated address masks are arranged in said binary CAM in a plurality of address entry groups, each address entry group having a respective mask number, with said address entry group having the highest mask number being located at the highest location of the CAM and said address entry group having the lowest mask number being located at the lowest location of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. The rejection of claims 56-87 under 35 U.S.C. 102(a) as being anticipated by Music Semiconductor Application Note AN-N22 "A Method* For Fast IPv4 CIDR Address Translation and Filtering Using the MUSIC WidePort LANCAM, LANCAM, and LANCAM 1st Family," dated January 1998 (the "LANCAM Publication") (attached as Exhibit 1 in Appendix B).

B. The rejection of claims 56-87 under 35 U.S.C. 102(a) as being anticipated by Music Semiconductor Application Note AN-N25 "Fast IPv4 And IPv4 CIDR Address Translation and Filtering Using the MUAC™ Routing Coprocessor (RCP)*," dated January 1998 (the "MUAC Publication") (attached as Exhibit 2 in Appendix B).

VII. ARGUMENT

A. CLAIMS 56-87 ARE PATENTABLE OVER MUSIC SEMICONDUCTOR APPLICATION NOTE AN-N22 (THE LANCAM PUBLICATION).

The LANCAM publication is not available as prior art in the present application because the subject matter of the LANCAM publication that is relevant to the rejected claims ("Relevant Subject Matter") was derived from the Appellants, and because the Appellants actually invented the Relevant Subject Matter.

1. The Feldmeier Declaration is sufficient evidence to disqualify the LANCAM publication as prior art.

Appellants submitted an Inventor's Declaration ("the Feldmeier Declaration") under 37 C.F.R. § 1.132 (attached as Exhibit 3 in Appendix B), signed by Appellant David C. Feldmeier on February 8, 2005. This sworn declaration unequivocally states that "the subject matter of the LANCAM publication relevant to the claims (Relevant Subject Matter) was derived from the Applicants," and "the Applicants actually invented the Relevant Subject Matter." This statement is sufficient to overcome the rejection because it consists of factual evidence of Appellants' position and it is uncontradicted by any other evidence.

The Final Office Action states, at page 9, that the Feldmeier Declaration is

insufficient to overcome the rejection because "it consists of mere allegation, and provides no factual evidence nor an unequivocal statement supporting the claims of authorship of LANCAM and MUAC or if said were published on their behalf, nor how publications were derived from the Applicants. (Please see MPEP 716.10.)" The Advisory Action mailed June 10 makes the same statement on page 2, and asserts on page 3 that "[t]he declaration of 2/15/05 is not a "submission of **evidence** establishing a **fact**" as required by 715.01(c)(II)" (emphasis in original). Appellants respectfully submit that the Advisory Action's misinterprets the law regarding inventor declarations under Rule 1.132.

"'[A]ffidavits traversing grounds of rejection' . . . are permitted under Rule 132, and may be considered as evidence." *In re Facius*, 408 F.2d 1396, 1352 (C.C.P.A. 1969). Specifically, the 35 U.S.C. §102(a) rejection can "be overcome by submission of a specific declaration by the applicant establishing that the article is describing applicant's own work." MPEP §2132.01. The Final Office Action and Advisory Action are incorrect in arguing that a sworn, unequivocal declaration submitted under Rule 1.132 is not evidence. The Feldmeier Declaration is a sworn statement that the two Appellants of the present application were also co-authors of the LANCAM publication. This unequivocal statement is not contradicted by any other evidence; the Final Office Action and Advisory Action do not allege that any contradictory evidence exists.

2. <u>An affidavit from co-inventor Tyler Arnold is not required to overcome the rejection.</u>

The Final Office Action also suggests at page 9 that "if the Affiant and Mr. Arnold are indeed the sole authors, that Mr. Arnold also submit a declaration to such facts mentioned above." Applicants responded on June 2, 2005, that a declaration by Mr.

Arnold is unnecessary because the Feldmeier Declaration is unequivocal and uncontradicted. The Advisory Action responded incorrectly by stating on page 3 that "affidavits by the other authors are required, as described in 2132.01" because the Feldmeier Declaration "does not indicate that Mr. Feldmeier is the sole inventor, as described in 715.01(c)(I)." This is "clear error." *In re Katz*, 687 F.2d 450, 455.

MPEP §2132.01 states that

> where the applicant is one of the co-authors of a publication cited against his or her application, the publication may be removed as a reference by the filing of affidavits made out by the other authors establishing that the relevant portions of the publication originated with, or were obtained from, applicant. Such affidavits are called disclaiming affidavits. *Ex parte irschler*, 110 USPQ 384 (Bd. App. 1952). The rejection can also be overcome by submission of a specific declaration by the applicant establishing that the article is describing applicant's own work. *In re Katz*, 687 F.2d 450, 215 USPQ 14 (CCPA 1982). However, if there is evidence that the co-author has refused to disclaim inventorship and believes himself or herself to be an inventor, applicant's affidavit will not be enough to establish that applicant is the sole inventor and the rejection will stand. *Ex parte Kroger*, 219 USPQ 370 (Bd. Pat. App. & Int. 1982) (emphasis added) (discussed below).

Disclaiming affidavits are not required where there is no contradictory evidence that casts doubt on a sworn declaration's statements of fact. *In re Katz*, 687 F.2d at 455 (It is "clear error" to hold that "disclaiming affidavits or declarations by the other authors are required to support applicant's position that he is, in fact, the sole inventor of the subject matter."). A disclaiming affidavit is merely one of many ways to show that a publication is derived from the Appellants.

Further, MPEP §715.01(c)(I) is inapplicable because Mr. Feldmeier is not the sole inventor and both authors of the LANCAM publication are also co-inventors of the Application. MPEP 715.01(c)(I). cited by the Advisory Action, clearly states that "[w]here the applicant is one of the co-authors of a publication cited against his or her application, . . . the applicant may overcome the rejection by filing a specific affidavit or declaration under 37 CFR 1.132 establishing that the article is describing applicant's own work. An affidavit or declaration by applicant alone indicating that applicant is the sole inventor and that the others were merely working under his or her direction is sufficient to remove the publication as a reference under 35 U.S.C. 102(a). *In re Katz*, 687 F.2d 450, 215 USPQ 14 (CCPA 1982)." The portion of MPEP 715.01(c)(I) discussing the problem of co-authors who are not also inventors is irrelevant here because all of the authors of the LANCAM publication, i.e. Mr. Feldmeier and Mr. Arnold, are also co-inventors of the Relevant Subject Matter.

As MPEP §2132.01 points out, "[t]he rejection can *also* be overcome by submission of a specific declaration by the applicant establishing that the article is describing applicant's own work." (emphasis added). "What is required is a reasonable statement supporting the basis for the applicant's position." *In re Katz*, 687 F.2d at 455. Where there is an uncontested sworn declaration and no evidence to the contrary, as with the Feldmeier Declaration, the Final Office Action and Advisory Action "should not have engaged in further speculation as to whether appellant's view was shared by his coauthors but rather should have accepted" the Feldmeier Declaration. *Id.*

Appellants respectfully submit that the Feldmeier Declaration filed February 15, 2005, establishes that the relevant subject matter of the LANCAM publication was derived

from the Appellants. Accordingly, the LANCAM publication is not prior art under 35 U.S.C. § 102(a). Therefore, the rejection of claims 56-87 under 35 U.S.C. § 102(a) as being anticipated by the LANCAM publication should be reversed.

B. CLAIMS 56-87 ARE PATENTABLE OVER MUSIC SEMICONDUCTOR APPLICATION NOTE AN-N25 (THE MUAC PUBLICATION).

The MUAC publication is also not available as prior art in the present application because the subject matter of the MUAC publication that is relevant to the rejected claims ("Relevant Subject Matter") was derived from the Appellants, and because the Appellants actually invented the Relevant Subject Matter.

1. The Feldmeier Declaration is sufficient to disqualify the MUAC publication as prior art.

As discussed above, Appellants submitted an Inventor's Declaration ("the Feldmeier Declaration") under 37 C.F.R. § 1.132 (attached as Exhibit 3 in Appendix B), signed by Appellant David C. Feldmeier on February 8, 2005. This sworn declaration unequivocally states that "the subject matter of the MUAC publication relevant to the claims (Relevant Subject Matter) was derived from the Applicants," and "the Applicants actually invented the Relevant Subject Matter." This statement is sufficient to overcome the rejection because it consists of uncontradicted evidence of Appellants' position.

The same arguments concerning the sufficiency of the Feldmeier Declaration made above with respect to the LANCAM publication also apply to the MUAC publication. That is, as detailed in the Feldmeier declaration, the LANCAM and MUAC publications were authored by Appellants Feldmeier and Arnold, and the Relevant Subject Matter of both publications were derived from the Appellants.

For at least the these reasons, Applicants respectfully submit that the Feldmeier Declaration filed February 15, 2005, establishes that the relevant subject matter of the MUAC publication was derived from the Appellants. Accordingly, the MUAC publication is not prior art under 35 U.S.C. § 102(a). Therefore, the rejection of claims 56-87 under 35 U.S.C. § 102(a) as being anticipated by the MUAC publication should be reversed.

VIII. CONCLUSION

For the reasons given above it is respectfully submitted that the final rejection of claims 56-87 is improper. Accordingly, Appellants requests reversal of all rejections by this honorable Board.

Dated: September 16, 2005

Respectfully submitted,

By _____

Thomas J. D'Amico
Registration No. 28,371
Jerome A. DeLuca
Registration No. 55,106
DICKSTEIN SHAPIRO MORIN &
OSHINSKY LLP
2101 L Street NW
Washington, DC 20037-1526
(202) 785-9700
Attorneys for Appellants

APPENDIX A – CLAIMS INVOLVED IN THE APPEAL

Claims 1- 55. (canceled)

56. A binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said binary CAM comprising a plurality of entries wherein:

each entry comprises:

(1) a first value comprising the logically ANDed communication system address and its associated mask; and

(2) a second value comprising the logically ANDed complement of said communication system address and its associated mask; and

wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

57. The binary CAM of claim 56 wherein entries having the most amount of contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

58. The binary CAM of claim 57 wherein said first value comprises n bits and said second value comprises n bits and wherein each one of said bits in said first value has an associated bit in said second value, said each one of said bits and said associated bit forming a binary pair, said binary pair representing one bit of said address as two bits in said CAM.

59. The binary CAM of claim 58 wherein a 1 in said ternary hierarchical address is represented as 10 in said CAM, wherein a 0 in said ternary hierarchical address is represented as 01 in said CAM, and wherein a don't care in said ternary hierarchical address is represented as 00 in said CAM.

60. The binary CAM of claim 58 wherein said first value is stored in an upper portion of said entry and said second value is stored in a lower portion of said entry.

61. The binary CAM of claim 60 wherein said entry comprises 64 bits and wherein n is 32.

62. A binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said binary CAM comprising a plurality of entries wherein:

each entry comprises a two bit representation for each bit in said ternary hierarchical address; and

wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

63. The binary CAM of claim 62 wherein:

(a) a 1 in said ternary hierarchical address is represented by said two bit representation in said CAM as 10;

(b) a 0 in said ternary hierarchical address is represented by said two bit representation in said CAM as 01; and

(c) a don't care in said ternary hierarchical address in represented by said two bit representation in said CAM as 00.

64. The binary CAM of claim 63 wherein entries having the most amount of contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

65. A method for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said method comprising the steps of:

logically ANDing each communication system address and its associated mask to form a first value;

logically ANDing the complement of each communication system address and its associated address mask to form a second value; and

storing said first value and said second value in an entry in said binary CAM based on the number of contiguous ones in said address mask.

66. The method of claim 65 wherein said step of storing said first value and said second value at a location comprises arranging entries such that entries having the most contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

67. The method of claim 66 wherein said first value comprises n bits and said second value comprises n bits and wherein said step of storing said first value and said second value comprises associating each one of said bits in said first value with one bit in said

second value to form a binary pair, said binary pair representing one bit of said address as two bits in said CAM.

68. The method of claim 67 wherein said step of representing one bit of said ternary hierarchical address as two bits in said CAM comprises:

(a) utilizing 10 in said CAM to represent a 1 in said ternary hierarchical address;

(b) utilizing 01 in said CAM to represent a 0 in said ternary hierarchical address; and

(c) utilizing 00 in said CAM to represent a don't care in said ternary hierarchical address.

69. The method of claim 67 wherein said step of storing said first value and said second value comprises storing said first value in an upper portion of said entry and storing said second value in a lower portion of said entry.

70. The method of claim 69 wherein said entry comprises 64 bits and wherein said step of storing said first value and said second value comprises storing said first value in the upper 32 bits and storing said second value in the lower 32 bits.

71. A method of storing for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said method comprising the steps of:

(a) generating a CAM entry for each ternary hierarchical address by:

(1) identifying that portion of each ternary hierarchical address that are don't cares;

(2) representing each bit in said ternary hierarchical address using bits in said CAM wherein:

(i) a 1 in said ternary hierarchical address is represented as 10 in said CAM;

(ii) a 0 in said ternary hierarchical address is represented as 01 in said CAM; and

(iii) a don't care in said ternary hierarchical address is represented as 00 in said CAM;

(b) positioning each CAM entry in said CAM based on the number of contiguous ones in said associated mask.

72. The method of claim 71 wherein said step of positioning each CAM entry in said CAM comprises arranging entries such that entries having the most contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

73. A method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address entry comprising a communication system address and an associated communication system address mask, said entries of ternary hierarchical addresses being stored in said binary CAM as a first value and a second value, said first value comprising the communication system address logically ANDed with said address mask and said second value comprising the complement of said communication system address logically ANDed with said address mask, and wherein all of said entries are ordered in said binary CAM based on the number of contiguous ones in said mask, said method comprising the steps of:

(a) loading a first register with the communication system address to be searched along with the complement of the communication system address;

(b) loading a second register with the communication system address to be searched along with the complement of the communication system address;

(c) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM;

(d) determining whether a bit match occurs between corresponding bits of said first register and each entry in said binary CAM as qualified by said corresponding bit of said second register; and

(e) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

74. The method of claim 73 wherein said step of determining whether a bit match occurs between corresponding bits of said first register and each entry in said binary CAM as qualified by said corresponding bit of said second register comprises declaring a bit match between corresponding bits of said first register and each entry in said binary CAM:

(1) if the corresponding bit in said second register is a 1; or

(2) if the corresponding bit in said second register is a 0 and the corresponding bits of said first register and each entry in said binary CAM are identical.

75. The method of claim 74 wherein said steps of loading a first and second register comprises loading the communication system address to be searched in an upper portion of said registers and loading said complement of the communication system address in a lower portion of said registers.

76. The method of claim 75 wherein said first and second registers comprise 64 bits and wherein said steps of loading said first and second registers comprises loading the communication system address to be searched in the upper 32 bits of said registers and loading said complement of the communication system address in said lower 32 bits of said registers.

77. A method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address comprising a communication system address and an associated communication system address mask, each ternary hierarchical address entry being stored in said binary CAM whereby 10 is stored in said entry for every 1 in said ternary hierarchical address, 01 is stored in said entry for every 0 in said ternary hierarchical address, and 00 is stored in said entry for every don't care in said ternary hierarchical address, and wherein all of said entries are ordered in said binary CAM based on the number of contiguous ones in said mask, said method comprising the steps of:

(a) loading a first register and second register with the communication system address to be searched by:

(1) loading a 10 in said first and second registers for every 1 in the ternary hierarchical address to be searched;

(2) loading a 01 in said first and second registers for every 0 in the ternary hierarchical address to be searched;

(3) loading a 00 in said first and second registers for every don't care in the ternary hierarchical address to be searched;

(b) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM;

(c) declaring a bit match between corresponding bits of said first register and each entry in said binary CAM:

(1) if the corresponding bit in said second register is a 1; or

(2) if the corresponding bit in said second register is a 0 and the corresponding bits of said first register and each entry in said binary CAM are identical; and

(d) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

78. A method for maintaining a sorted CAM to enable longest matches in a single search cycle when hierarchical addresses are added to, or deleted from, the CAM in a communication system utilizing hierarchical addresses and associated address masks, said method comprising the steps of:

(a) segmenting the CAM into blocks wherein each block corresponds to a single hierarchical mask and wherein said blocks are arranged in the CAM such that

the lowest CAM addresses contain the highest hierarchical masks and the highest CAM addresses contain the lowest hierarchical masks;

(b) storing hierarchical addresses according to said block having a corresponding hierarchical mask; and

(c) tracking the first address and the next free address of each of said blocks and the size of each of said blocks.

79. A content addressable memory (CAM) of a communications system utilizing ternary hierarchical addressing and associated address masks, said CAM comprising:

a plurality of address entries and associated address masks that are arranged in said CAM by mask number, with address entries having the highest mask number being located at address entry locations at the top of the CAM and address entries having the lowest mask number being located at address entry locations at the bottom of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

80. An apparatus for storing a plurality of address entries and associated address masks in a communication system utilizing ternary hierarchical addressing and associated address masks, said apparatus comprising:

a binary CAM;

a binary-encoded ternary converter coupled to said binary CAM wherein said binary-encoded ternary converter converts each ternary value into a corresponding binary-encoded ternary value to form said address entries; and

wherein said plurality of address entries and associated address masks are arranged in said binary CAM in a plurality of address entry groups, each address entry group having a respective mask number, with said address entry group having the highest mask number being located at the highest location of the CAM and said address entry group having the lowest mask number being located at the lowest location of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

81. A method for accelerating the routing of hierarchical addressing in a communication system which utilizes ternary hierarchical addressing and associated address masks, said method comprising the steps of:

(a) obtaining communication system hierarchical addresses and associated masks to form a plurality of address entries; and

(b) storing said plurality of address entries in a content-addressable memory (CAM) device by mask number wherein said mask number is defined as the number of contiguous ones in an associated address mask and wherein address entries having the highest mask number are stored in address entry locations at the highest location of the CAM and address entries having the lowest mask number are stored at address entry locations at the lowest location of the CAM.

82. A content addressable memory (CAM) of a communication system as defined in claim 79 wherein said plurality of address entries comprises:

a first group of address entries sharing a first said address mask and a second group of address entries sharing a second said address mask, said first and second groups being located at different locations of the CAM.

83. A content addressable memory (CAM) of a communication system as defined in claim 82 further comprising at least one vacant address entry location disposed within said CAM between said first and second groups of address entries.

84. An apparatus for storing a plurality of address entries and associated address masks in a communication system as defined in claim 80 further comprising:

at least one vacant address entry location within said CAM adjacent at least one of said address entry groups.

85. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 81 further comprising the steps of:

receiving an address value in a comparand register of said CAM;

matching a further plurality of addresses, including a subset of said communication system hierarchical addresses, to said address value; and

outputting an output hierarchical address of said further plurality of addresses according to respective storage locations of said further plurality of addresses.

86. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 85 wherein said output hierarchical address of said further plurality has a storage location lowest among said respective storage locations of said further plurality of addresses.

87. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 85 wherein said further plurality of said communication system hierarchical addresses each includes at least one identical bit.

APPENDIX B – EVIDENCE

Exhibit 1: The LANCAM Publication: Music Semiconductor Application Note AN-N22 "A Method* For Fast IPv4 CIDR Address Translation and Filtering Using the MUSIC WidePort LANCAM, LANCAM, and LANCAM 1st Family"

Exhibit 2: The MUAC Publication: Music Semiconductor Application Note AN-N25 "Fast IPv4 And IPv4 CIDR Address Translation And Filtering Using the MUAC™ Routing Coprocessor (RCP)*"

Exhibit 3: The Feldmeier Declaration: Declaration of David C. Feldmeier under 37 CFR 1.132

MUSIC SEMICONDUCTORS

Application Note AN-N22

A Method* For Fast IPv4 And IPv4 CIDR Address Translation And Filtering Using The MUSIC WidePort LANCAM®, LANCAM®, AND LANCAM® 1ST Family

*The methods in this application note are patent pending.

INTRODUCTION

The Internet Protocol is used to transmit packets of digital data over the Internet. Packet filtering is a complex task and is typically handled by specialized computers known as routers and layer 3 switches. As the demand for bandwidth increases, methods to increase the performance of these routers must be found. A task fundamental to the performance of IP is the filtering of the 32-bit IPv4 addresses used to identify the source and destination of each packet.

This application note describes a technique that provides fast searches and reduces address table size. The technique takes advantage of IP Classless Inter Domain Routing (CIDR) to find the so-called "longest match" in an IP routing table. The MUSIC WidePort LANCAM family is used here, but other MUSIC LANCAMs can use the same method, resulting in various price/performance ratios.

CIDR

Formerly, IP addresses were divided into separate classes: Class A, B, C, D, and E, which enumerate the range of addresses. However, these class distinctions proved to be wasteful of address resources and led to large IP address tables.

CIDR is a common method of reducing address table size, which categorizes address aggregations on arbitrary mask bit boundaries. IPv6 will take this scheme further with 128-bit addresses. CIDR reduces the size of IP tables by making it possible to group thousands of entries under one entry in the CIDR table. Each address entry is associated with a network mask in a CIDR address table. Figure 1 shows two examples of how the 32-bit network mask values are structured according to Internet Request For Comments 940 and 1519. The masks always have a structure consisting of contiguous ones on the left and contiguous zeros on the right. A mask will never have ones and zeros interleaved or zeros on the left and ones on the right. Figure 2 shows examples of invalid mask structures.

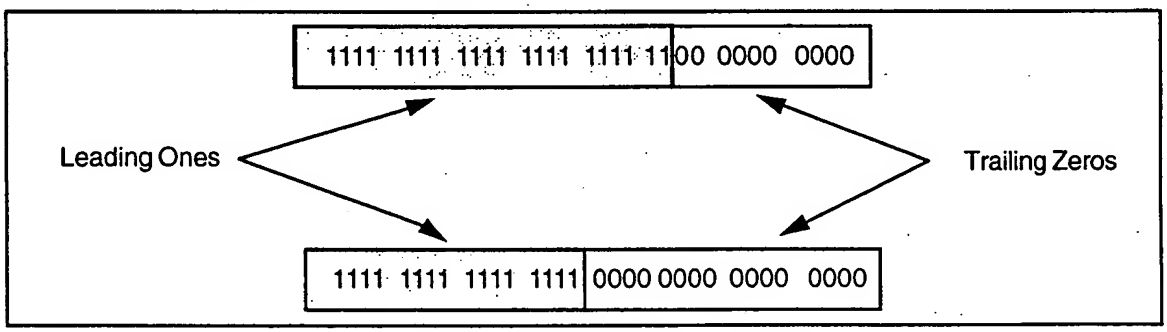

Figure 1: Structure of Valid Network Masks

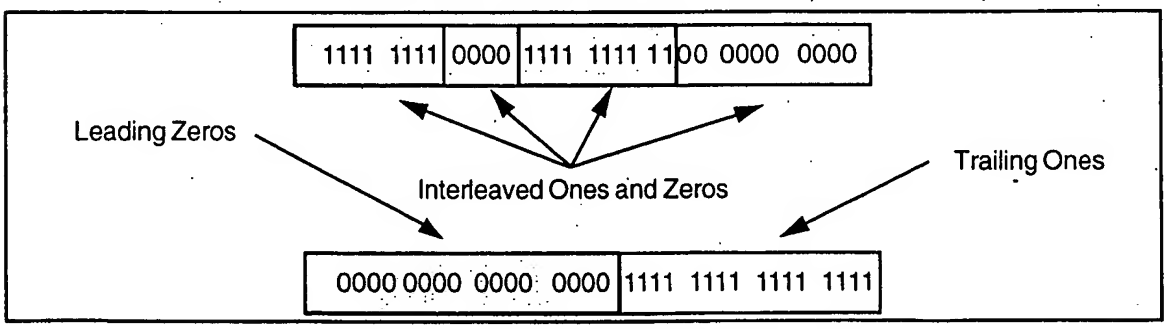

Figure 2: Structure of Invalid Network Masks

Address	Mask
C0.18.0C.15	FF.FF.FF.FF/32
C0.18.0C10	FF.FF.FF.F0/28
C0.18.0C.00	FF.FF.FC.00/22
C0.18.00.00	FF.FF.00.00/16

Table 1: Addresses and Masks from a CIDR Routing Table

The CIDR address ranges are a subset of IPv4 Class C addresses, which are in the range 192.0.0.0 to 223.255.255.255. By assigning the addresses in this manner, a router running IPv4 CIDR can hold addresses with CIDR boundaries in addition to addresses with Class A, B, C, D, and E boundaries. A router running only IPv4 will ignore the CIDR boundaries and treat a CIDR address as a group of Class C addresses. To ensure that IPv4–only routers can operate with IPv4 CIDR routers, the CIDR routers must "explode" their CIDR address aggregations when communicating with the non-CIDR routers. Table 1 shows four entries from a CIDR routing table (in hexadecimal format) and the network masks that relate to them. It can be seen that the mask boundaries are clearly demarcated. The number after the backslash indicates the number of contiguous ones in the mask. For example, the first entry's mask is FF.FF.FF.FF, which corresponds to 32 bits set to 1. This mask is an IP mask, where a 1 indicates a "care" location and a 0 indicates a "don't care." A "care" bit means that the corresponding bit locations of the numbers being compared must be identical. A "don't care" bit indicates that the bit locations are not used in the comparison. The more the number of 1-bits or "cares" in a mask, the less the number of possible matches. Therefore, a higher mask number is more precise than a lower mask number. Note, an IP address search may match multiple CIDR routing table entries because masking is used to further specify smaller address ranges.

In Table 1, the last address/mask entry actually corresponds to 1100 0000 0001 1000 XXXX XXXX XXXX XXXX where the X's stand for "don't cares." Any search value whose top 16 bits equal 1100 0000 0001 1000 would match the last entry. Note that in Table 1, the entries are ordered by mask value from most 1s (FF.FF.FF.FF) to the entry with the least number of 1s (FF.FF.00.00 in this case). A search for C0.18.0C.15 would yield matches on all the entries but the first entry is chosen because it is the match with the longest group of 1s in the mask (the so-called "longest match"). A search for C0.18.0C.16 would match only the second, third, and fourth entries.

USING CAM TO IMPLEMENT CIDR

The basic principal that makes CAM ideal for implementing IPv4 CIDR is the ability to encode ternary values. A ternary value is such that each element can encode 0, 1, or X where X matches both 0 and 1. CIDR tables consist of a hierarchical structure of addresses partitioned by IP masks. A technique can be used that will encode a ternary value and store it in a CAM. To store a ternary value in a binary CAM, the CAM width must be twice the address width. The MUSIC WidePort 64-bit LANCAM family is ideal for IPv4 and IPv4 CIDR. To perform CIDR filtering using a LANCAM, it is necessary to store each address and mask value in separate memory array locations while using a management routine running from a CPU or possibly an ASIC.

When multiple matches occur during a search, CAMs select the "best" entry by their physical addresses. Matches with lower addresses are selected over matches with higher addresses. For example, if a search yields matches at CAM address 1 and 2, the match at address 1 is selected. In a typical layer 2 address filtering application, the order of the entries in the CAM is unimportant because it is expected that there will be no more than one match result from any given search.

However, hierarchical address structures like IPv4, IPv4 CIDR, and IPv6 differ by having the possibility of multiple matches during a search operation. The best match is defined as the "longest match," or the entry whose mask has the largest number of contiguous "cares" or 1s. Therefore, if a CAM is to be used to store hierarchical addresses, sorting is necessary. There are two alternatives for this sorting: sorting when inserting entries or sorting when reading matches. Fast hardware lookup for high bandwidth routing and switching places an emphasis on high-speed searches. Choosing to sort when inserting entries is therefore a logical decision because it enables fast constant time address resolution.

To place hierarchical entries in the CAM, it is necessary to sort their position in the CAM so that the longest match is selected first. Since CAMs select the best match by lowest CAM address value, the entries with the longest match mask are stored in the lowest CAM addresses. If the following condition is met, a hierarchical address search is guaranteed always to resolve to the longest match entry in one CAM search cycle.

For all CAM entries n: **mask(n) >=mask(n+1)**

Ternary	Binary Encoded
0	01
1	10
X	00

Table 2: Binary Encoding of Ternary Values

The management routine runs an algorithm that ensures that the entries in the CAM are sorted properly so that searches always find the longest matching entry (i.e., the entry with the most number of 1s in its mask). By ensuring that the CAM entries are always ordered, the longest matching address is always found within a single compare operation. The time to insert an entry depends upon how much sorting is done and is explained later.

BINARY ENCODED TERNARY

To facilitate IPv4 CIDR, a CAM is split into separate groups or blocks demarcated by IP mask values. If a group of addresses that share the same mask are to be stored, there must be some way of encoding the "don't care" condition indicated by the address mask. Therefore, two binary CAM bits are required to encode a single ternary (1, 0, X) entry. To binary encode the ternary values (1, 0, X), 64 bits is required to store a 32-bit value.

The basic principle is that the binary value and its complement are stored if 1 or 0 is required, but when the "don't care" condition is needed, 0 and 0 is stored. Table 2 shows how the three ternary values are encoded.

The same principle is used if we need to search the CAM for a ternary value. We also have to incorporate a Mask register when the compare is performed. To search the CAM memory array for the ternary value 1, we would load 10 into both the Comparand register and the Mask register. The search would yield the results shown in Table 3. The ternary equivalent of each value is shown in bold after the binary encoded value.

The search resulted in the CAM entries 00 (X) and 10 (1) matching the comparand 10 (1). The comparand is effectively seen as X0 during the compare because the Mask register is also loaded with 10. This makes the first bit of the binary encode value a "don't care" and will therefore match with either 1 or 0. Similarly, a search for 01 (0) would produce a match with the CAM entries 01 (0) and 00 (X). This effectively gives a match result when there is a ternary X value, which is what is expected.

When the Mask register is used to mask out bits during the compare, it effectively allows ternary "don't cares" to be incorporated during a search. The Mask register works in the opposite way from an IP mask. When the register bit is set to 1, the result will always be a match condition. Table 4 shows the Truth Table of CAM compares using the Mask register.

The same principle is used to store 32-bit ternary values as 64-bit binary encoded ternary values in a 64-bit wide CAM. The high order bit of each 2-bit ternary value is stored in the upper 32 bits of the CAM entry while the low order bit is stored in the lower 32 bits of the CAM entry. For example, if we were storing C0.18.0C.15, bits 63:60 (hex C) of the CAM entry would be 1100 and bits 31:29 would be 0011. This is

Comparand bits	10 (1)	10 (1)	10 (1)
Mask bits	10	10	10
CAM bits	00 (X)	01 (0)	10 (1)
Result	match	no-match	match

Table 3: CAM Compare Truth Table with Ternary Values

Comparand bit	0	1	0	1	0	1	0	1
Mask bit	0	0	1	1	0	0	1	1
CAM bit	0	0	0	0	1	1	1	1
Result	match	no-match	match	match	no-match	match	match	match

Table 4: CAM Compare Truth Table

equivalent to encoding the ternary value 1100 as the binary encoded equivalent 10 10 01 01.

To form the groups of addresses with the same IP masks the individual addresses have their masks added to them as they are inserted into the CAM memory array. To store an address and mask in a 64-bit wide CAM location, the bit-wise logical AND of the address and mask are stored in the upper 32 bits while the bit-wise logical AND of the mask and the address's 1s complement are stored in the lower 32 bits. Table 5 shows how the address C0.18.0C.15 and mask FF.FF.FF.FF would be stored in a CAM memory location.

There is enough information in each entry to determine the explicit mask/address pair that generated it. To search for the entry C0.18.0C.15 we would perform a compare using C0.18.0C.15 as the upper 32 bits and its 1s complement, 3F.E7.F3.EA, as the low 32 bits. A Mask register also is loaded with the same value and is used when the compare takes place. The CAM Mask register is used to determine which bits take part in the search and which bits are effectively ignored. As seen in Table 4, the CAM mask works in the opposite way from an IP mask. This means that a 1 corresponds to a "don't care" while a 0 corresponds to a "care" bit.

Each address is logical ANDed with the appropriate mask value before being stored as binary encoded ternary. To perform a search operation using this encoding method, the address being searched for is copied to the upper 32 bits of both the Comparand and Mask registers and the 1s complement of the address is copied to the lower 32 bits.

This again is loading the binary encoded ternary equivalent. Table 6 shows how four 32-bit ternary values and their IP masks would be encoded and stored in a 64-bit CAM.

Because of the ability to binary encode the ternary value X ("don't care"), the IP address and IP mask can be combined and stored as 64 bits. These 64-bit values are stored in the CAM memory array in IP mask order. That means the IP mask values that have the higher number of contiguous 1s are stored in the higher priority locations. The section on sorting CIDR entries gives a full explanation of how the entries are sorted into blocks.

If IP address C0.18.0C.15 were encoded in the normal manner and compared with the four entries in Table 6, each entry would produce a match result. If the entries were sorted properly, the highest priority match would be entry number 1, which would be the desired result. Table 7 shows the results of the comparison between C0.18.0C.15 and entry number 3. This shows how the combination of the binary encoded ternary values and the use of the Mask register work to produce a match result.

The MUSIC Semiconductors WidePort LANCAM has a 32-bit I/O bus. This allows the 64-bit binary encoded ternary values to be loaded into the Mask register in two cycles. The search procedure consists of the following:

1. Two Short cycles – Load IP Address to bits[63:32] of Mask register and load NOT (IP Address) to bits [31:0] of Mask register.

2. One Short cycle – Copy the Mask register to the Comparand register.

63 32	31 0
ADDR AND MASK	**(!ADDR) AND MASK**
C0.18.0C.15	3F.E7.F3.EA

Table 5: How Address and Mask Values are Stored in CAM

Entry	IP Address	IP Mask	CAM bits [63:32]	CAM bits [31:0]
1	C0.18.0C.15	FF.FF.FF.FF	C0.18.0C.15	3F.E7.F3.EA
2	C0.18.0C.10	FF.FF.FF.F0	C0.18.0C.10	3F.E7.F3.E0
3	C0.18.0C.00	FF.FF.FC.00	C0.18.0C.00	3F.E7.F0.00
4	C0.18.00.00	FF.FF.00.00	C0.18.00.00	3F.E7.00.00

Table 6: IP Address and Mask Stored as Binary Coded Ternary Values

Comparand register	1100 0000 0001 1000 0000 1100 0001 0101 0011 1111 1110 0111 1111 0011 1110 1001
Mask register	1100 0000 0001 1000 0000 1100 0001 0101 0011 1111 1110 0111 1111 0011 1110 1001
CAM entry	1100 0000 0001 1000 0000 1100 0000 0000 0011 1111 1110 0111 1111 0000 0000 0000
Result (see notes)	xxyy yyyy yyyx xyyy yyyy xxyy yyyx yxyx yyxx xxxx xxxy yxxx xxxx yyxx xxxy xyyx

NOTES:

1. x = The result was a "don't care" match.
2. y = The result was a normal binary match.

Table 7: Results of 64-bit Comparison

3. One Long cycle – Initiate a comparison with the CAM memory array through the Mask register.

4. One Medium cycle – Read out the CAM array address of highest-priority match. This address would be used as an index to access associated data stored in external RAM.

Using these five cycles, the search time required is as follows:

50ns speed grade CAM: 210ns with a 15ns clock, or 4.76 million searches per second.

70ns speed grade CAM: 280ns with a 20ns clock, or 3.57 million searches per second.

This IPv4 CIDR method also can be used with the LANCAM and LANCAM 1ST families. Table 8 shows the comparison among MUSIC LANCAMs used for IPv4 CIDR.

SORTING THE CIDR ENTRIES

If all the entries in the CAM array have been sorted correctly, searching for entries is trivial. The task of maintaining the correct order of entries is performed by the insertion and management routines. A solution to managing this task is to partition the CAM into RAM addressable blocks for ease in sorting. To ensure that the longest match is always the highest-priority match, each block is associated with a specific IP mask and only entries with the corresponding IP mask are placed in that block. For a 32-bit mask, there are at most 32 partitions. Figure 3 on page 7 shows an example with a 100 entry CAM partitioned into four blocks, each possible of holding 25 entries. Each entry is in the form described earlier with the address & IP mask in the upper 32 bits of the CAM location and (! Address) & IP mask in the lower 32 bits.

By sorting the entries in the CAM according to their mask, it is guaranteed that a search will always return the longest matching address because the longest match entries are stored in lower CAM addresses. It is unnecessary to sort the CAM entries within each mask block, because all the entries within that mask block must have the exact same mask. Therefore, the number of blocks created to partition the CAM must equal the number of different masks that will be stored in the CAM. For example, it is incorrect to partition the CAM into four segments corresponding to the IP masks FF.FF.FF.FF, FF.FF.FF.00, FF.FF.00.00 and FF.00.00.00 and then store any entries with masks between FF.FF.FF.FF and FF.FF.FF.00 in the top block. If this situation is allowed to exist, the highest-priority match pulled from the CAM will not necessarily be the longest match. By ensuring that only entries with the same mask are stored in each block, no more than one match can occur in that block. If multiple matches occur in the same block and the blocks are sorted properly, it can only mean that there are multiple copies of the same entry in that block.

Each CAM partition has a set of pointers and flags associated with it that indicate its status:

Floor *	- This pointer indicates the first address in the block.
Blocksize	- This indicates the number of entries in the block.
Nxtfree *	- This pointer indicates the next address to place an entry.
Backpressure flag	- This indicates this and all blocks "below" are filled. Required for recursive sorting algorithm.

The database is initialized with the required number of blocks and all the pointers set to the correct values. In the example shown in Figure 3, the Floor* and Nxtfree* of Block 0 would both be set to physical address 0, while the Floor*

Device	Wideport LANCAM X485A/L - 50	LANCAM FAMILY X480A/L – 70	LANCAM 1ST X481L - 10
Time To Search	195ns	340ns	480ns
Searches Per Sec.	5.13 Million	2.94 Million	2.08 Million

Explanation of Timing

WidePort LANCAM (32-bit I/O) Using a 15ns clock

Two short cycles to load 64-bit Binary Encoded Ternary address into Mask register	=	60
One short cycle to copy Mask register to Comparand register	=	30
One long cycle to initiate compare of comparand with memory array (through Mask register)	=	60
One medium cycle to read Status register to give address index for external associated data	=	45
Total		**195 ns**

LANCAM Family (16-bit I/O) Using a 20ns clock

Four short cycles to load 64-bit Binary Encoded Ternary address into Mask register	=	160
One short cycle to copy Mask register to Comparand register	=	40
One long cycle to initiate compare of comparand with memory array (through Mask register)	=	80
One medium cycle to read Status register to give address index for external associated data	=	60
Total		**340 ns**

LANCAM 1ST (16-bit I/O) Using a 20ns clock

Four short cycles to load 64-bit Binary Encoded Ternary address into Mask register	=	240
One short cycle to copy Mask register to Comparand register	=	60
One long cycle to initiate compare of comparand with memory array (through Mask register)	=	100
One medium cycle to read Status register to give address index for external associated data	=	80
Total		**480 ns**

Table 8: Comparison of Search Operation Timing Using IPv4 CIDR

and Nxtfree* of Block 1 is set to 24 (100/4 –1), if we decide to partition the CAM into equal blocks. In a real application, it would make sense to initialize the block sizes according to the IP mask. A mask of FF.00.00.00 could only have 255 entries in the block, while a mask of FF.80.00.00 could have twice as many, and so on.

Inserting A New Entry

Figure 4 shows the case when the easiest insertion occurs. This is when the destination block is not full and there is therefore an empty entry space available. The binary encoded ternary value can be written into the next free location of the block. If the destination block is full, the pointers must be altered and entries at the edges of the blocks must be moved.

Insertion into a full block involves some pointer and entry manipulation. Figure 5 shows the procedure required for inserting an entry into a full block. Three blocks are shown each with their respective floor pointers (F0, F1, and F2). A new entry has to be inserted into block 0, but it is full and therefore unable to accept any new data. Block 1 has some free space, which can be manipulated and moved to Block 0.

The Floor (or first) entry of the next block (pointed by F1) is copied to the location pointed to by the next free entry pointer (NF1) of that block. This will free a space at the beginning of Block 1. F1 is then incremented to its new value, while the blocksize of B1 is decreased because it is giving up space to B0. NF0 and NF1 are also incremented to show what their respective new next free locations will be once the insertion is complete. The new entry is then inserted at the newly created space in Block 0, which was formerly the Floor entry of Block 1. When the insertion process requires manipulation between two blocks in this way the following CAM operations are required:

1. One cycle to read the entry that has to be moved from the original F1 location.

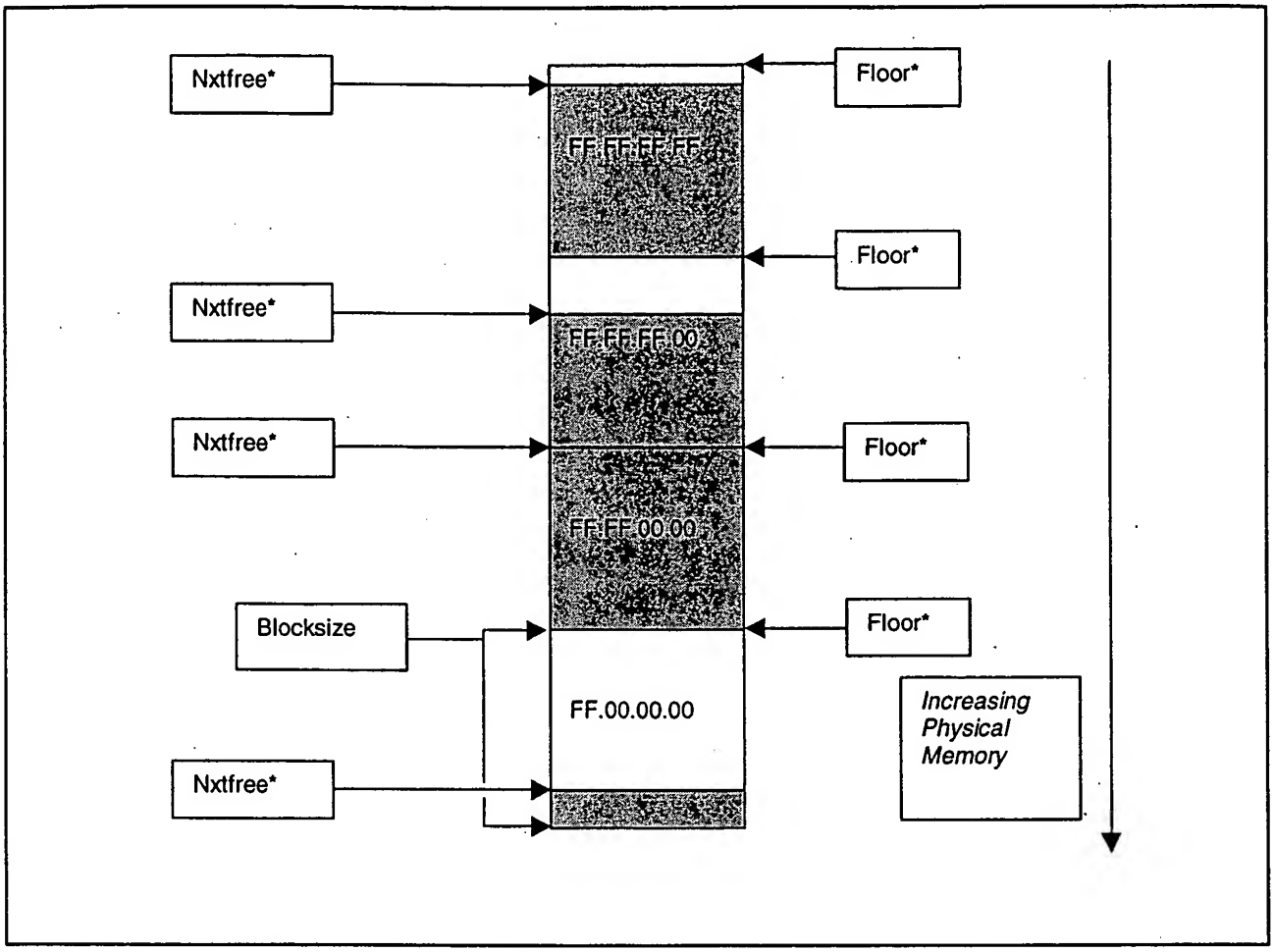

Figure 3: CAM Array Partitioned into Blocks

2. One cycle to write this entry into the next free space in the block.
3. One cycle to write the new entry into the newly creates space.

It is possible for the next block to be full; therefore a recursive shuffling operation must occur. Figure 6 shows how the recursive shuffling is performed to enable a new entry to be placed in Block 0. In (A), the new entry is destined for Block 0. Blocks 0, 1, and 2 are all full, but Block 3 has one free entry left. Therefore the free entry in Block 3 must be shuffled up to be located in Block 0. Stage (B) shows the floors of the blocks being copied to the subsequent lower blocks. First the Floor entry of Block 3 is moved to the next free location of the same block. Next the Floor of Block 2 is copied downward into the newly free location, after which the Floor of the Block 1 likewise moves down. Finally the new entry can be moved into the newly created space in Block 0. Stage (C) shows the new entry located in Block 0 with all the floor pointers updated.

This method ensures that none of the entries are removed from the CAM while the sorting is occurring. This is important if the processor were updating the table in real time while high bandwidth searches were occurring. The processor insertion routine could then be interrupted by a search and queue its operations up while it waits for lulls in search accesses to continue. Only the new IP address is inaccessible to the searches during this period. The database stored in the CAM will always give the longest match and will always be valid.

Deleting Entries

Another important operation in maintaining the accuracy of the sorted IP table is the deletion of entries. To ensure that the memory that is made available from a delete

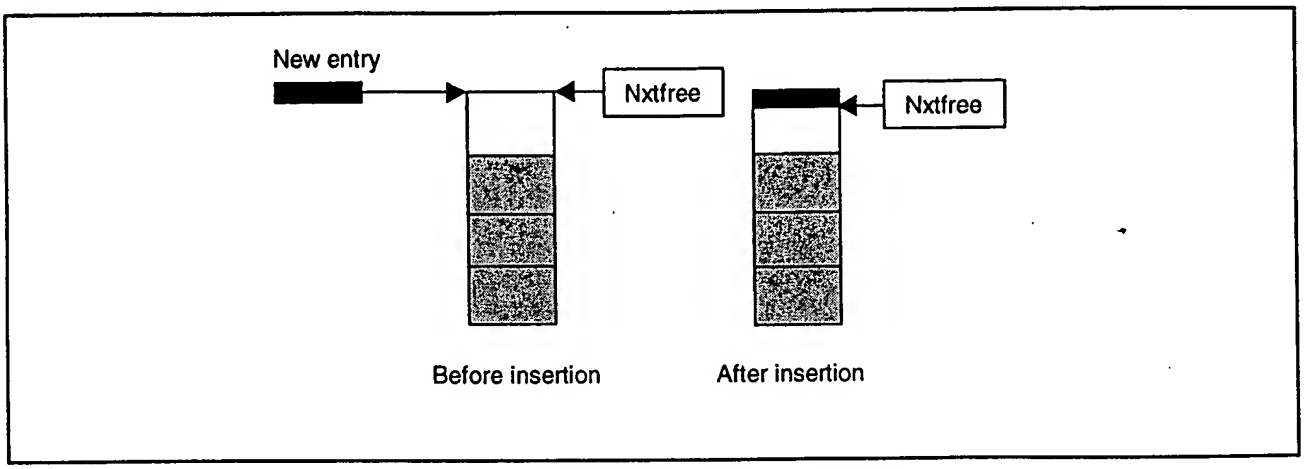

Figure 4: Insertion into an Empty Entry Space

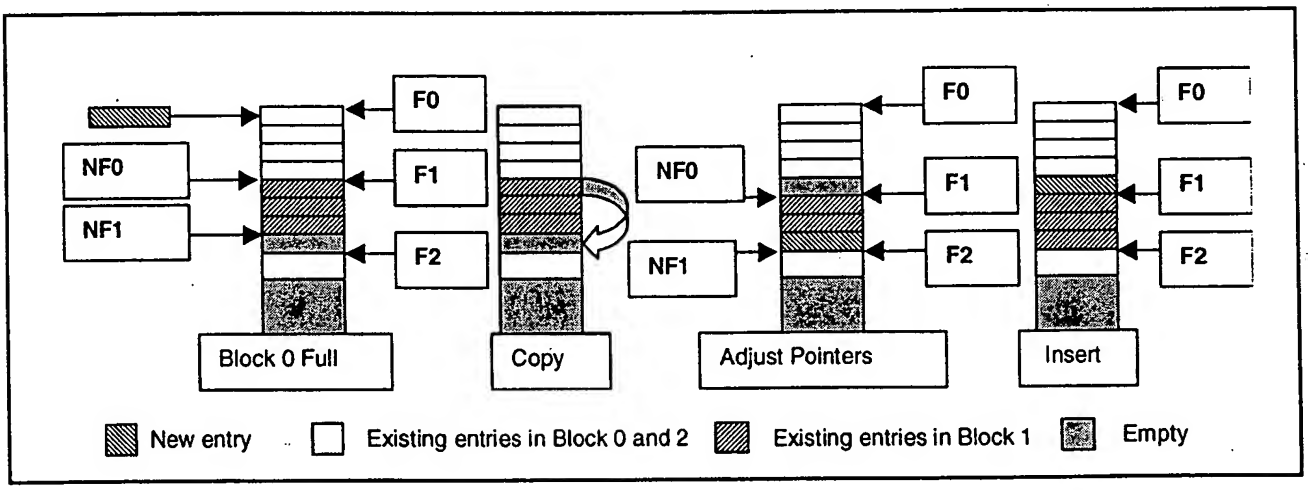

Figure 5: Insertion into a Full Block

operation is reused, the block in which the deletion occurred must be reorganized. The delete operation is shown in Figure 7. Block 1 (B1) contains an entry that has to be deleted. Copying the last entry in the block over the entry that is to be deleted does this. The previous location of the last entry is then invalidated. The pointer that points to the next free location in the block is then adjusted to point to the correct entry.

Searching For Explicit CAM Entries

Depending upon how the routing table updates are performed, it may be necessary to search for explicit IP address/mask combinations to determine if an entry is already in the CAM. In this case, the longest match is not what is being looked for. Therefore, the search procedure is slightly different from the one used to locate the longest match. When searching for the longest match, the Comparand register contents are duplicated in a Mask register and the search is performed with the relevant bits masked out.

When searching for explicit entries there is no need to use the Mask register. The (addr & mask), (!addr & mask) representation of the IP address/mask pair is loaded into the Comparand register and a compare is initiated. The address/mask pair is compared with the entries in the CAM memory array. This search will only yield a match if the CAM entry exactly matches the entry in the Comparand register. The LANCAM family has foreground and background register sets, which can make this task easier.

The foreground registers should be set to do loading and searches for the longest matches as described earlier in this Application Note while the background registers should be set to perform explicit CAM searches that would perform compares without using a Mask register. It is then only necessary to switch between the two register sets as needed.

Sorting Upwards

A special case to consider is when an entry is to be added to a full block and the blocks below are also full. When this occurs, entries cannot be swept further "downwards" to produce an empty location in the required block. Instead, they must be pushed up if there is any free space above.

A "Backpressure" flag, mentioned earlier, is set in the last block when there is no more space available. The meaning of the "backpressure flag" is this: if it is not set, there is at least one more free entry somewhere between the current block and the highest physical memory location. If it is set, then all blocks below the current block are full. Blocks attempting to push downward check this flag and, if it is set, are forced to try to push entries upward. In this case, recursion downward into the blocks would not work because there is no more free memory. Instead, recursion must go upwards. The same sorting routine is applied here with the difference being that the sorting goes in the opposite direction. Figure 8 shows an example of recursive sorting upwards.

In A, a new entry is destined for the last block but it is full, requiring the edge sorting to go upward instead of downward. Secondly, B shows the moves and inserts that will be done to free an entry space in the final block. As before, the Routing database stored in the CAM will always yield a correct search result due to the order of the moves and inserts. Finally, C shows the blocks as represented by their readjusted pointers and block size values.

Interleaving Operations

It is possible to interleave operations of the CAM in order to keep the table updated while continuing to perform high bandwidth searches. For searches, the atomic operation is the five instruction load/load/copy/search/read index sequence. The insertion operation can be broken up into different atomic operations that can be interleaved in between gaps in the atomic search operations. The following enumerate the different atomic insert operations:

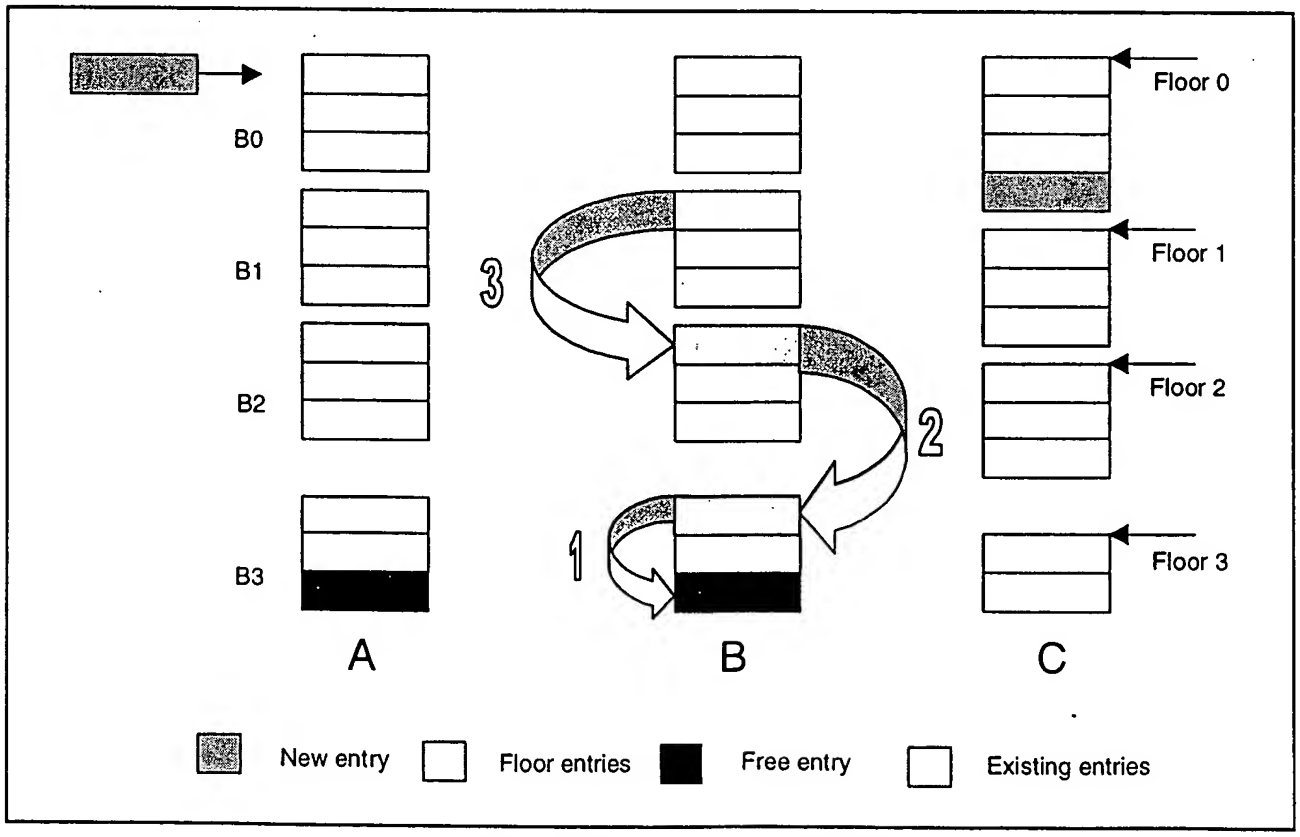

Figure 6: Recursive Insertion into Lower Blocks

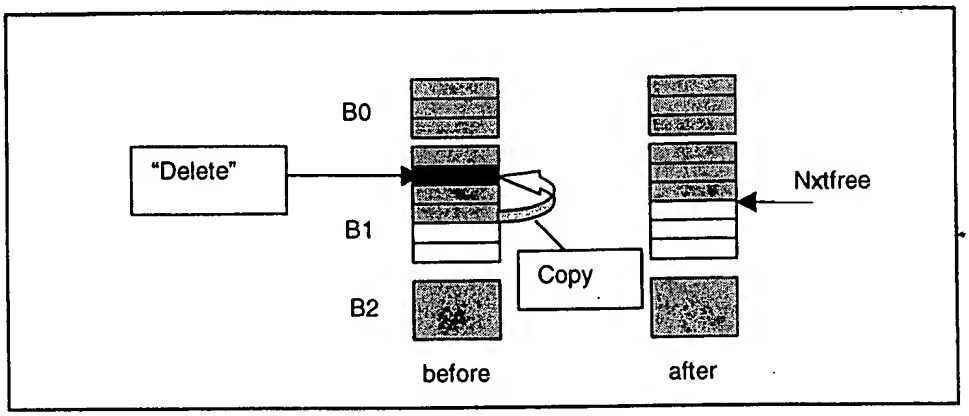

Figure 7: Deleting an Entry

1. Insert to address:
   Two short cycles : load (addr & mask) and ((!addr) & mask)
   One short cycle : insert to address

2. Read out entry to be copied/moved

Figure 9 shows how the four inserts and three copies required in the previous example on recursive sorting could be queued and interleaved in between address searches. The worst case number of cycles for this can be calculated assuming that the new entry is inserted to the first block but all blocks except the last are full. For *k* partitions, there are:

k-1 CAM read cycles
k CAM write cycles
k RAM read/write cycles

An important point to note is that since *k* entries are moved, the overhead of moving their associated data in RAM must also be considered. However, this worst case only happens once; with thoughtful initial block allocation, inserting a new entry into the CAM will take one write cycle. A good precautionary measure to avoid long sorting operations is to adjust the block size and spacing during idle periods to avoid shuffling during busy periods.

If the associated data entries stored in RAM are more than a few bytes in length, a scheme using a second section of RAM to incorporate pointers might be considered. The pointers would point to the associated data entries of each CAM entry, which would avoid moving large blocks of data. When the associated data entries are shuffled, it is a simple case of rearranging the pointers. This would remove the need to move the associated data around, which will reduce the associated data maintenance time.

ROUTING TABLE UPDATES

The method described in this Application Note is flexible enough to be used with different styles of updating the routing tables and therefore allows a system designer or software engineer the ability to use the most suitable method available. Using the algorithm described in this Application Note in conjunction with dynamic routing table updates would be more efficient than reloading the routing table in the CAM and refilling it. This is because search operations could still be performed during the updates.

CONCLUSION

Because of the ability to encode ternary values in a binary CAM, the MUSIC Semiconductors CAMs provide a high-speed and flexible alternative to software routing tables. By incorporating IPv4 CIDR in the manner described in this Application Note, table sizes can be greatly reduced and hence reduce component costs as compared with a more conventional IPv4 table.

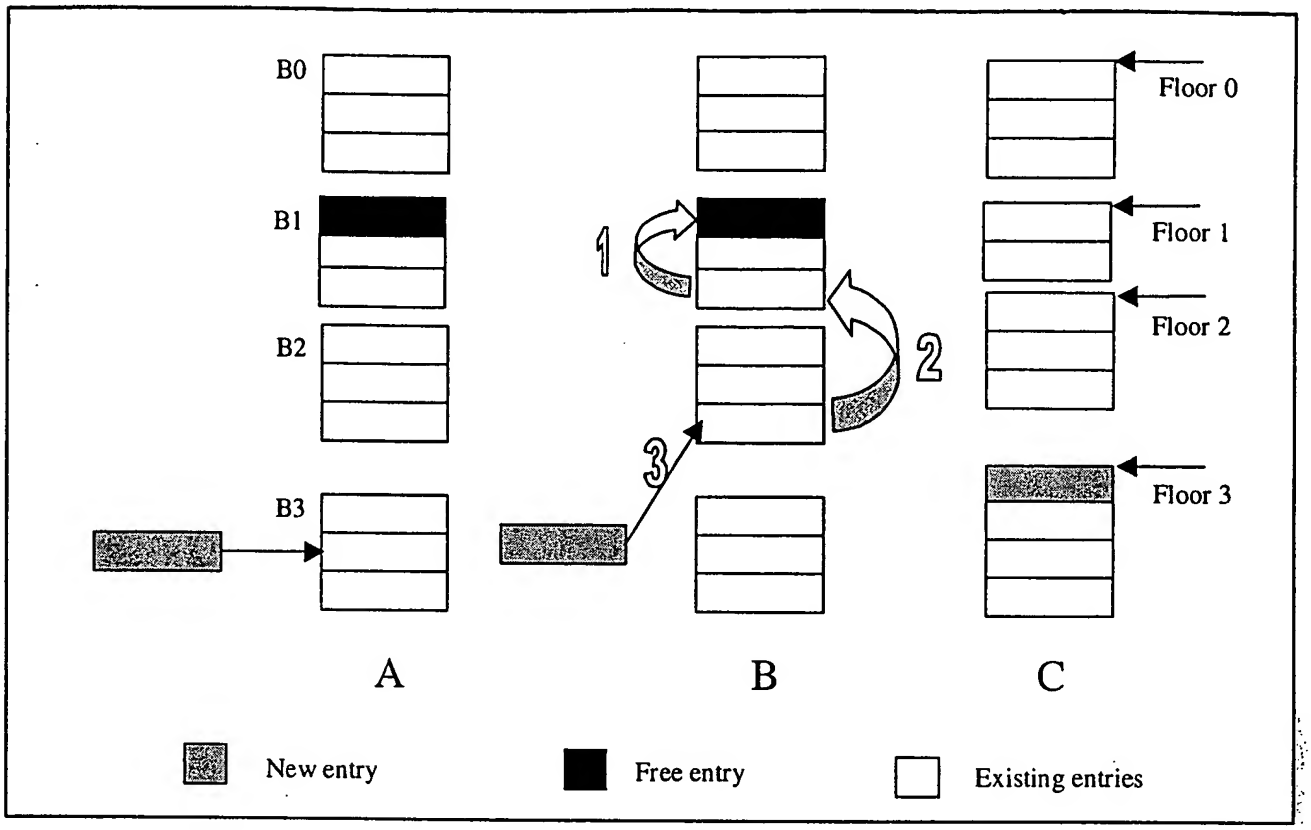

Figure 8: Recursive Sorting Upwards

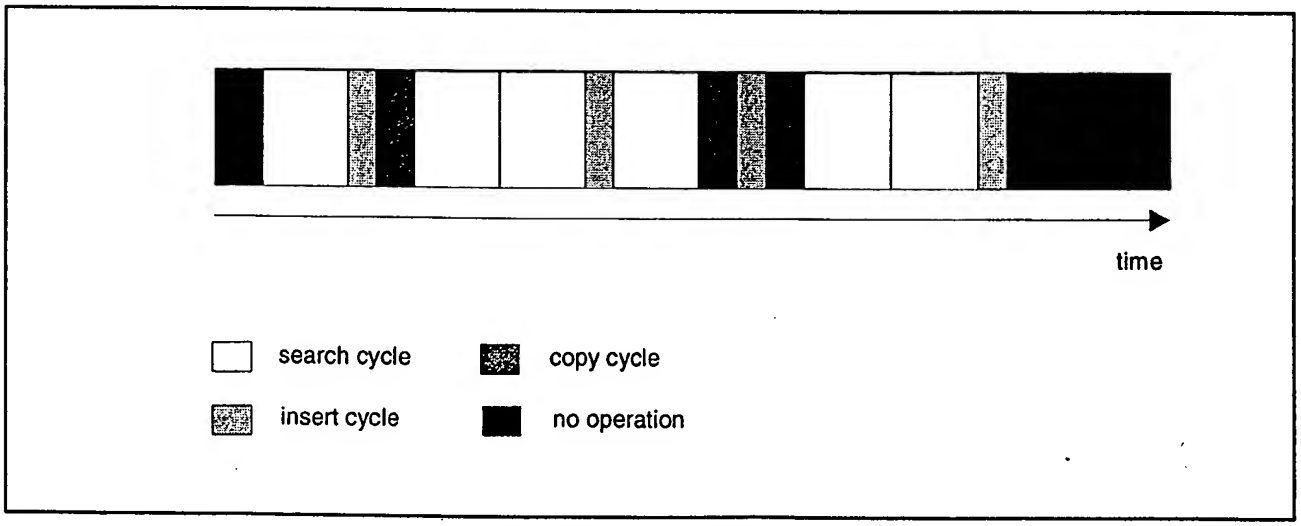

Figure 9: Cycle Interleaving

NOTES

MUSIC Semiconductors Agent or Distributor:

MUSIC SEMICONDUCTORS

http://www.music.com
email: info@music.com

USA Headquarters
MUSIC Semiconductors
254 B Mountain Avenue
Hackettstown, New Jersey 07840
USA
Tel: 908/979-1010
Fax: 908/979-1035
USA Only: 800/933-1550 Tech. Support
888/226-6874 Product Info.

Asian Headquarters
MUSIC Semiconductors
Special Export Processing Zone 1
Carmelray Industrial Park
Canlubang, Calamba, Laguna
Philippines
Tel: +63 49 549 1480
Fax: +63 49 549 1023/1024
Sales Tel/Fax: +632 723 62 15

European Headquarters
MUSIC Semiconductors
Torenstraat 28
6471 JX Eygelshoven
Netherlands
Tel: +31 45 5462177
Fax: +31 45 5463663

MUSIC SEMICONDUCTORS

Application Note AN-N25

Fast IPv4 And IPv4 CIDR Address Translation And Filtering Using The MUAC™ Routing CoProcessor (RCP)*

*The methods in this Application Note are patent pending.

INTRODUCTION

The Internet Protocol is used to transmit packets of digital data over the Internet. Packet recognition is a complex task typically handled by specialized computers known as routers and layer 3 switches. As the demand for bandwidth increases, methods to increase the performance of these routers must be found. A task fundamental to the performance of IP is the identification of the 32-bit IPv4 addresses used to identify the source and destination of each packet.

This Application Note describes how to use the MUAC Routing CoProcessor (RCP) for its fundamental task. The MUAC RCP is able to provide fast searches and to reduce address table size. The MUAC RCP takes advantage of IP Classless Inter Domain Routing (CIDR) to find the so-called "longest match" or "best prefix match" in an IP routing table. The MUAC RCP also can be used to build a Multi-Layer Search engine because it can contain a mixture of addresses of a different types. In addition to IPv4 addresses it is fairly easy to process MAC addresses and/or IP Multicasts and even IPX. Using the technique described in MUSICs Application Note AN-N27 Using 64-Bit Wide MUSIC CAMs and RCPs for IP Flow Recognition, the MUAC RCP also can be used for Layer 4 lookups. The MUAC RCP is available in various speed grades. These devices are fast enough to guarantee multi-gigabit address processing at wire speed. Although not developed within this Application Note, similar approaches would allow the MUAC RCP to be used for IP/ATM purposes.

CIDR

Formerly, IP addresses were divided into separate classes: Class A, B, C, D, and E, which enumerate the range of addresses. However, these class distinctions proved to be wasteful of address resources and led to large IP address tables.

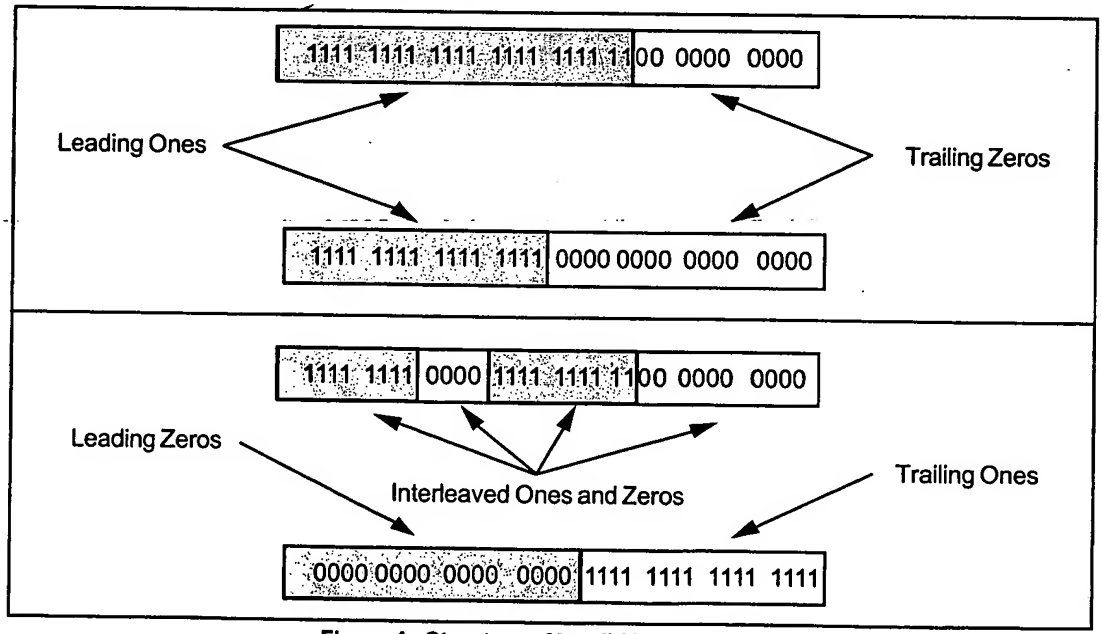

Figure 1: Structure of Invalid Network Masks

Address	Mask
C0.18.0C.15	FF.FF.FF.FF/32
C0.18.0C10	FF.FF.FF.F0/28
C0.18.0C.00	FF.FF.FC.00/22
C0.18.00.00	FF.FF.00.00/16

Table 1: Addresses and Masks from a CIDR Routing Table

CIDR is a common method of reducing address table size, which categorizes address aggregations on arbitrary mask bit boundaries. IPv6 will take this scheme further with 128-bit addresses. CIDR reduces the size of IP tables by making it possible to group thousands of entries under one entry in the CIDR table. Each address entry is associated with a network mask in a CIDR address table. Figure 1 shows two examples of how the 32-bit network mask values are structured according to Internet Request for Comments 940 and 1519. The masks always have a structure consisting of contiguous ones on the left and contiguous zeros on the right. A mask will never have ones and zeros interleaved or zeros on the left and ones on the right. Figure 1 shows examples of invalid mask structures.

The CIDR address ranges are a subset of IPv4 Class C addresses, which are in the range 192.0.0.0 to 223.255.255.255. By assigning the addresses in this manner, a router running IPv4 CIDR can hold addresses with CIDR boundaries in addition to addresses with Class A, B, C, D, and E boundaries. A router running only IPv4 will ignore the CIDR boundaries and treat a CIDR address as a group of Class C addresses. To ensure that IPv4, (only routers can operate with IPv4 CIDR routers) the CIDR routers must "explode" their CIDR address aggregations when communicating with the non-CIDR routers. Table 1 shows four entries from a CIDR routing table (in hexadecimal format) and the network masks that relate to them. It can be seen that the mask boundaries are clearly demarcated. The number after the backslash indicates the number of contiguous ones in the mask. For example, the first entry's mask is FF.FF.FF.FF, which corresponds to 32 bits set to 1. This mask is an IP mask, where a 1 indicates a "care" location and a 0 indicates a "don't care." A "care" bit means that the corresponding bit locations of the numbers being compared must be identical. A "don't care" bit indicates that the bit locations are not used in the comparison. The more the number of 1-bits or "cares" in a mask, the less the number of possible matches. Therefore, a higher mask number is more precise than a lower mask number. Note, an IP address search may match multiple CIDR routing table entries because masking is used to further specify smaller address ranges.

In Table 1, the last address/mask entry actually corresponds to 1100 0000 0001 1000 XXXX XXXX XXXX XXXX where the Xs stand for "don't cares." Any search value whose top 16 bits equal 1100 0000 0001 1000 would match the last entry. Note that in Table 1, the entries are ordered by mask value from most 1s (FF.FF.FF.FF) to the entry with the least number of 1s (FF.FF.00.00 in this case). A search for C0.18.0C.15 would yield matches on all the entries but the first entry is chosen because it is the match with the longest group of 1s in the mask (the so-called "longest match"). A search for C0.18.0C.16 would match only the second, third, and fourth entries.

USING THE MUSIC MUAC RCP TO IMPLEMENT CIDR

The basic principal that makes MUSIC's MUAC RCP ideal for implementing IPv4 CIDR is the ability to encode not only binary, but also ternary values. A ternary value is such that each element can encode 0, 1, or X where X matches both 0 and 1. CIDR tables consist of a hierarchical structure of addresses partitioned by IP masks. A technique can be used that will encode a ternary value and store it in the RCP. To store a ternary value in a binary RCP, the RCP word width must be twice the address width. The MUAC RCP is ideal for IPv4 and IPv4 CIDR. To perform CIDR filtering using the MUAC RCP , it is necessary to store each address and mask value in separate memory array locations while using a management routine running from a CPU or possibly an ASIC. (MUSIC can provide the C language source-code for this typical management routine.)

When multiple matches occur during a search, the MUAC selects the "best" or "longest prefix" entry by their physical addresses. Matches with lower addresses are selected over matches with higher addresses. For example, if a search yields matches at addresses 1 and 2, the match at address 1 is selected. (Note: In a typical layer 2 address filtering application, the order of the entries in the address database is not important because it is expected that there will be no more than one match result from any given search.)

However, hierarchical address structures like IPv4, IPv4 CIDR, and IPv6 differ by having multiple matches during a search operation. The best match is defined as the "longest match," or the entry whose mask has the largest number of

Ternary	Binary Encoded
0	01
1	10
X	00

Table 2: Binary Encoding of Ternary Values

contiguous "cares" or 1s or exact bit matching. Therefore, if a RCP is to be used to store hierarchical addresses, sorting is necessary. There are two alternatives for this sorting: sorting when inserting entries or sorting when reading matches. Fast hardware lookup for high bandwidth routing and switching places an emphasis on high-speed searches. Choosing to sort when inserting entries is therefore a logical decision because it enables fast constant time address resolution.

To place hierarchical entries in the RCP, it is necessary to sort their position in the RCP so that the longest match is selected first. Since the RCP selects the best match by lowest address values, the entries with the longest match mask are stored in the lowest addresses. If the following condition is met, a hierarchical address search is guaranteed always to resolve to the longest match entry in one search cycle.

For all RCP words n: mask(n) >=mask(n+1)

The management routine runs an algorithm that ensures that the entries in the RCP are sorted properly so that searches always find the longest matching entry (for instance, the entry with the most number of 1s in its mask). By ensuring that the RCP entries are always ordered, the longest matching address always is found with a single compare operation. The time to insert an entry depends upon how much sorting is done and is explained later. Various techniques balance the best utilization of the CAM array with the shortest insertion time; selection would depend on the system nature.

BINARY ENCODED TERNARY

To facilitate IPv4 CIDR, the RCP address database is split into separate groups or blocks demarcated by IP mask values. If a group of addresses that share the same mask are to be stored, there must be some way of encoding the "don't care" condition indicated by the address mask. Therefore, two binary bits are required to encode a single ternary (1, 0, X) entry. To binary encode the ternary values (1, 0, X), 64 bits are required to store a 32-bit value.

The basic principle is that the binary value and its complement are stored if 1 or 0 is required, but when the "don't care" condition is needed, 0 and 0 is stored. Table 2 shows how the three ternary values are encoded.

The same principle is used if we need to search the RCP for a ternary value. We also use a Mask register when the compare is performed. To search the RCP memory array for the ternary value 1, we would load 10 into both the Comparand register and the Mask register. The search would yield the results shown in Table 3. The ternary equivalent of each value is shown in bold after the binary encoded value.

The search resulted in the entries 00 (X) and 10 (1) matching the comparand 10 (1). The comparand is effectively seen as X0 during the compare because the Mask register is also loaded with 10. This makes the first bit of the binary

Comparand bits	10 (1)	10 (1)	10 (1)
Mask bits	10	10	10
RCP bits	00 (X)	01 (0)	10 (1)
Result	match	no-match	match

Table 3: CAM Compare Truth Table with Ternary Values

Comparand bit	0	1	0	1	0	1	0	1
Mask bit	0	0	1	1	0	0	1	1
RCP bit	0	0	0	0	1	1	1	1
Result	match	no-match	match	match	no-match	match	match	match

Table 4: RCP Compare Truth Table

encode value a "don't care" and will therefore match with either 1 or 0. Similarly, a search for 01 (0) would produce a match with the entries 01 (0) and 00 (X). This effectively gives a match result when there is a ternary X value, which is expected.

When the Mask register is used to mask out bits during the compare, it effectively allows ternary "don't cares" to be incorporated during a search. The Mask register works in the opposite way from an IP mask. When the register bit is set to 1, the result will always be a match condition. Table 4 shows the Truth Table of RCP compares using the Mask register.

The same principle is used to store 32-bit ternary values as 64-bit binary encoded ternary values in a 64-bit wide CAM. The high order bit of each 2-bit ternary value is stored in the upper 32 bits of the RCP word while the low order bit is stored in the lower 32 bits of the RCP word. For example, if we were storing C0.18.0C.15, bits 31:29 (hex C) of the RCP word would be 1100 and bits 63:60 would be 0011. This is equivalent to encoding the ternary value 1100 as the binary encoded equivalent 10 10 01 01.

To form the groups of addresses with the same IP masks the individual addresses have their masks added to them as they are inserted into the RCP. To store an address and mask in a 64-bit RCP word, the bit-wise logical AND of the address and mask are stored in the lower 32 bits while the bit-wise logical AND of the mask and the address's one's complement are stored in the upper 32 bits. Table 5 shows how the address C0.18.0C.15 and mask FF.FF.FF.FF would be stored in a RCP word.

There is enough information in each entry to determine the explicit mask/address pair that generated it. To search for the entry C0.18.0C.15 we would perform a compare using C0.18.0C.15 as the lower 32 bits and its one's complement, 3F.E7.F3.EA, as the upper 32 bits. A Mask register also is loaded with the same value and is used when the compare takes place. The RCP Mask register is used to determine which bits take part in the search and which bits are effectively ignored. As seen in Table 4, the CAM mask works in the opposite way from an IP mask. This means that a 1 corresponds to a "don't care" while a 0 corresponds to a "care" bit.

Each address is logical ANDed with the appropriate mask value before being stored as binary encoded ternary. To perform a search operation using this encoding method, the address being searched for is copied to the lower 32 bits of both the Comparand and Mask registers and the one's complement of the address is copied to the upper 32 bits. This again is loading the binary encoded ternary equivalent. Table 6 shows how four 32-bit ternary values and their IP masks would be encoded and stored in a 64-bit wide RCP.

Because of the ability to binary encode the ternary value X ("don't care"), the IP address and IP mask can be combined and stored as 64 bits. These 64-bit values are stored in the RCP address database in IP mask order. That means the IP mask values that have the higher number of contiguous 1's are stored in the higher priority locations. The section on sorting CIDR entries gives a full explanation of how the entries are sorted into blocks.

63 ... 32	31 ... 0
(~ADDR) AND MASK	ADDR AND MASK
3F.E7.F3.EA	C0.18.0C.15

Table 5: How Address and Mask Values are Stored in RCP Entries

Entry	IP Address	IP Mask	CAM bits [31:0]	CAM bits [63:32]
1	C0.18.0C.15	FF.FF.FF.FF	3F.E7.F3.EA	C0.18.0C.15
2	C0.18.0C.10	FF.FF.FF.F0	3F.E7.F3.E0	C0.18.0C.10
3	C0.18.0C.00	FF.FF.FC.00	3F.E7.F0.00	C0.18.0C.00
4	C0.18.00.00	FF.FF.00.00	3F.E7.00.00	C0.18.00.00

Table 6: IP Address and Mask Stored as Binary Coded Ternary Values

Comparand register	0011 1111 1110 0111 1111 0011 1110 1010	1100 0000 0001 1000 0000 1100 0001 0101
Mask register	0011 1111 1110 0111 1111 0011 1110 1010	1100 0000 0001 1000 0000 1100 0001 0101
RCP word entry	0011 1111 1110 0111 1111 0000 0000 0000	1100 0000 0001 1000 0000 1100 0000 0000
Result (see notes)	yyxx xxxx xxxy yxxx xxxx yyxx xxxy xyxy	xxyy yyyy yyyx xyyy yyyy xxyy yyyx yxyx

NOTES:
1. x = The result was a "don't care" match.
2. y = The result was a normal binary match.

Table 7: Results of 64-bit Comparison

If IP address C0.18.0C.15 were encoded in the normal manner and compared with the four entries in Table 6, each entry would produce a match result. If the entries were sorted properly, the highest priority match would be entry number 1, which would be the desired result. Table 7 shows the results of the comparison between C0.18.0C.15 and entry number 3. This shows how the combination of the binary encoded ternary values and the use of the Mask register work to produce a match result.

The MUAC RCP can search for longest match in a single operation by using the special Ternary Compare command (CMPT DQ). The data, the 32bit IPv4 (CIDR) address, from the DQ bus is used as both bits 31–0 of the comparand and mask. The bit-wise complement of the data from the DQ bus is calculated inside the MUAC RCP and is used as both bits 63–32 of the comparand and mask. The resulting 64-bit comparand and mask are compared with all valid entries in the Memory array. So applying the 32 bit IPv4 address and executing the CMPT DQ command results in a ternary search of the memory to find the longest match. The address of the longest matching location and the Page address of the active device will become available on the output bus of the MUAC RCP, the PA:AA bus. The result on this bus can be used to directly index into external RAM. Since the PA:AA bus is pipelined the next search-operation can be started as soon as the result becomes available. This results in an average lookup time of one RCP Compare cycle.

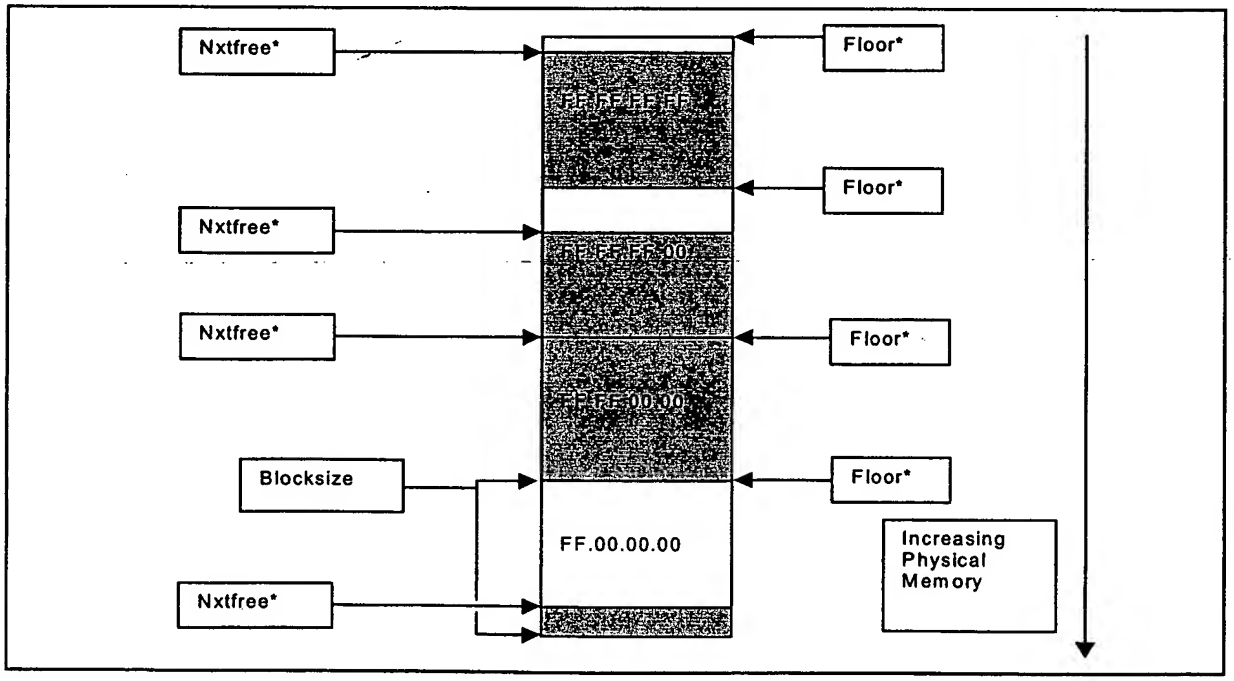

Figure 2: RCP Address Database Array Partitioned into Blocks

SORTING THE CIDR ENTRIES

If all the entries in the RCP address database have been sorted correctly, searching for entries is trivial. The task of maintaining the correct order of entries is performed by the insertion and management routines. A solution to managing this task is to partition the database into RAM addressable blocks for ease in sorting. To ensure that the longest match is always the highest-priority match, each block is associated with a specific IP mask and only entries with the corresponding IP mask are placed in that block. For a 32-bit mask, there are at most 32 partitions. Figure 2 shows an example with a 100 entry RCP partitioned into four blocks, each possible of holding 25 entries. Each entry is in the form described earlier with the address & IP mask in the lower 32 bits of the word and (~Address) & IP mask in the upper 32 bits.

By sorting the entries in the RCP address database according to their mask, it is guaranteed that a search will always return the longest matching address because the longest match entries are stored in lower word addresses. It is unnecessary to sort the RCP words within each mask block, because all the entries within that mask block must have the exact same mask. Therefore, the number of blocks created to partition the RCP must equal the number of different masks that will be stored in the RCP. For example, it is incorrect to partition the CAM into four segments corresponding to the IP masks FF.FF.FF.FF, FF.FF.FF.00, FF.FF.00.00 and FF.00.00.00 and then store any entries with masks between FF.FF.FF.FF and FF.FF.FF.00 in the top block. If this situation is allowed to exist, the highest-priority match pulled from the RCP will not necessarily be the longest match. By ensuring that only entries with the same mask are stored in each block, no more than one match can occur in that block. If multiple matches occur in the same block and the blocks are sorted properly, it can only mean that there are multiple copies of the same entry in that block.

Each RCP partition has a set of pointers and flags associated with it that indicate its status:

Floor *	-This pointer indicates the first address in the block.
Blocksize	-This indicates the number of entries in the block.
Nxtfree *	-This pointer indicates the next address to place an entry.
Backpressure flag	-This indicates this and all blocks "below" are filled. Required for recursive sorting algorithm.

The database is initialized with the required number of blocks and all the pointers set to the correct values. In the example shown in Figure 2, the Floor* and Nxtfree* of Block 0 would both be set to physical address 0, while the Floor* and Nxtfree* of Block 1 is set to 24 (100/4 –1), if we decide to partition the RCP address database into equal blocks. In a real application, it would make sense to initialize the block sizes according to the IP mask. A mask of FF.00.00.00 could only have 255 entries in the block, while a mask of FF.80.00.00 could have twice as many, and so on.

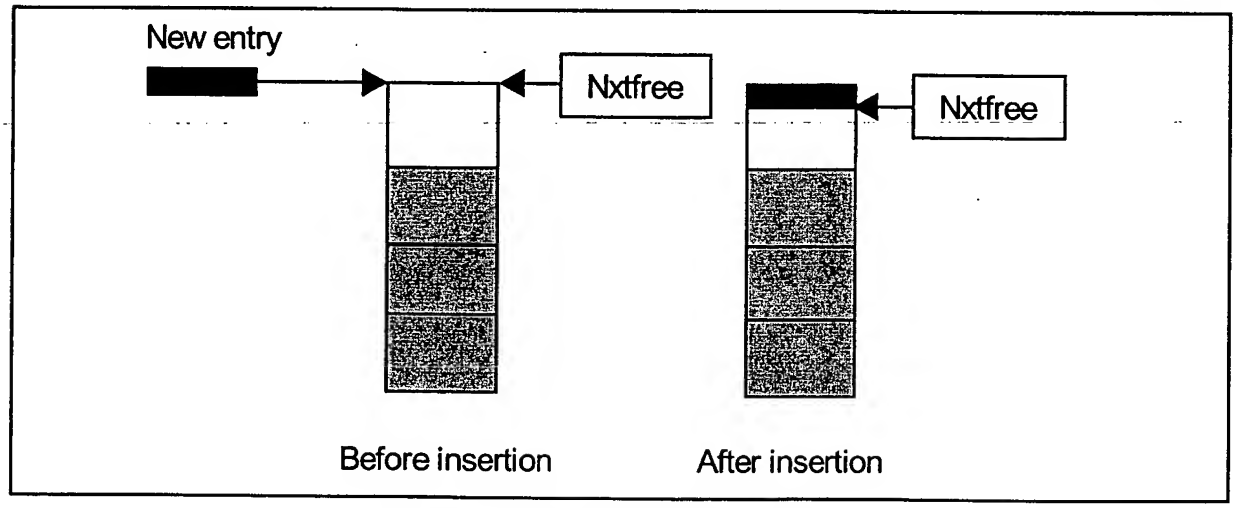

Figure 3: Insertion into an Empty Entry Space

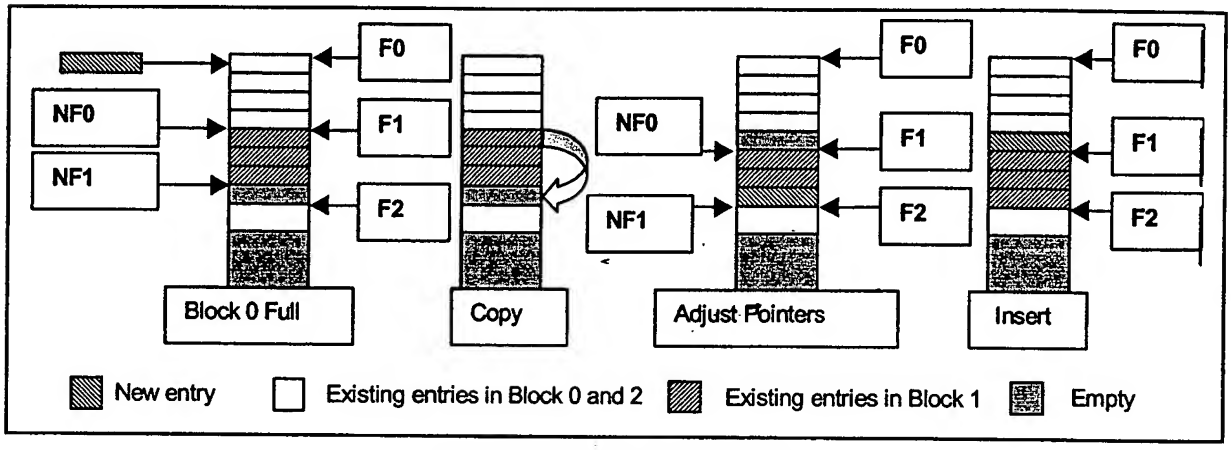

Figure 4: Insertion Into a Full Block

Inserting A New Entry

Figure 3 shows the case when the easiest insertion occurs. This is when the destination block is not full and there is therefore an empty entry space available. The binary encoded ternary value can be written into the next free location of the block. The commands are as follows:

WRL [aaa]
WRH [aaa]
(Code descriptions can be found in Figure 10.)

Where aaa is the physical address of the next free location of the block.

If the destination block is full, the pointers must be altered and entries at the edges of the block must be removed.

Insertion into a full block involves some pointer and entry manipulation. Figure 4 shows the procedure required for inserting an entry into a full block. Three blocks are shown each with their respective floor pointers (F0, F1, and F2). A new entry has to be inserted into block 0, but it is full and therefore unable to accept any new data. Block 1 has some free space, which can be manipulated and moved to Block 0.

The Floor (or first) entry of the next block (pointed by F1) is copied to the location pointed to by the next free entry pointer (NF1) of that block. This will free a space at the beginning of Block 1. F1 is then incremented to its new value, while the blocksize of B1 is decreased because it is giving up space to B0. NF0 and NF1 are also incremented to show what their respective new next free locations will be once the insertion is complete. The new entry is then inserted at the newly created space in Block 0, which was formerly the Floor entry of Block 1. When the insertion process requires manipulation between two blocks in this way the following CAM operations are required:

1. Two short cycles to move the entry that has to be moved from the original F1 location into a temporary register (CR).
   WR AR {MR000}
   MOV CR, [AR]

2. Two short cycles to move this entry into the next free location in the block.
   WR AR {MR000}
   MOV [AR], CR

3. Two short cycles to write the new entry into the newly created space.
   WRL [aaa]
   WRH [aaa]

(Code descriptions can be found in Figure 10.)

It is possible for the next block to be full; therefore a recursive shuffling operation must occur. Figure 5 shows how the recursive shuffling is performed to enable a new entry to be placed in Block 0. In (A), the new entry is destined for Block 0. Blocks 0, 1, and 2 are all full, but Block 3 has one free entry left. Therefore the free entry in Block 3 must be shuffled up to be located in Block 0. Stage (B) shows the floors of the blocks being copied to the subsequent lower

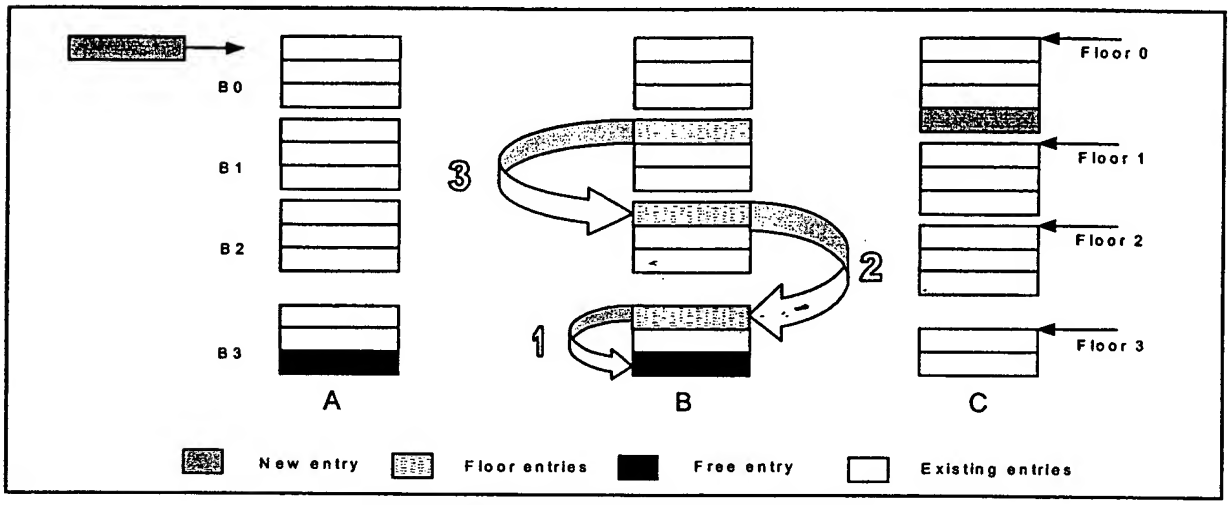

Figure 5: Recursive Insertion into Lower Blocks

blocks. First the Floor entry of Block 3 is moved to the next free location of the same block. Next the Floor of Block 2 is copied downward into the newly free location, after which the Floor of the Block 1 likewise moves down. The new entry can be moved into the newly created space in Block 0. Stage (C) shows the new entry located in Block 0 with all the floor pointers updated.

This method ensures that none of the entries are removed from the RCP while the sorting is occurring. This is important if the processor was updating the table in real time while high bandwidth searches were occurring. The processor insertion routine could then be interrupted by a search and queue its operations up while it waits for lulls in search accesses to continue. Only the new IP address is inaccessible to the searches during this period. The database stored in the RCP will always give the longest match and will always be valid. To insert entries into the MUAC RCP the following instructions have to be executed. External to the MUAC RCP the 64-bit value of ((~addr)&mask), (addr&mask) is calculated. Then the (addr&mask) value is written into the lower part (bits 31–0) of the location defined by the address value present on the AC bus. This address is the value of the next free location of the block. Then the ((~addr)&mask) value is written into the higher part of the same location. The /VB input must be low in order to set the new entry valid.

Deleting Entries

Another important operation in maintaining the accuracy of the sorted IP table is the deletion of entries. To ensure that the memory that is made available from a delete operation is reused, the block in which the deletion occurred must be reorganized. The delete operation is shown in Figure 6. Block 1 (B1) contains an entry that has to be deleted. Copying the last entry in the block over the entry that is to be deleted does this. The previous location of the last entry is then invalidated. The pointer that points to the next free location in the block is then adjusted to point to the correct entry. To find the entry that has to be deleted a search for an explicit RCP word has to be executed. The commands for deleting an explicit entry are as follows:

```
WRL     CR      {MR000}
CMPWH   DQ      {MR000}
RST     V@[HPM]
```

(Code descriptions can be found in Figure 10.)

Searching For Explicit RCP Entries

Depending upon how the routing table updates are performed, it may be necessary to search for explicit IP address/mask combinations to determine if an entry is already in the RCP. In this case, the longest match is not what is being looked for. Therefore, the search procedure is slightly different from the one used to locate the longest

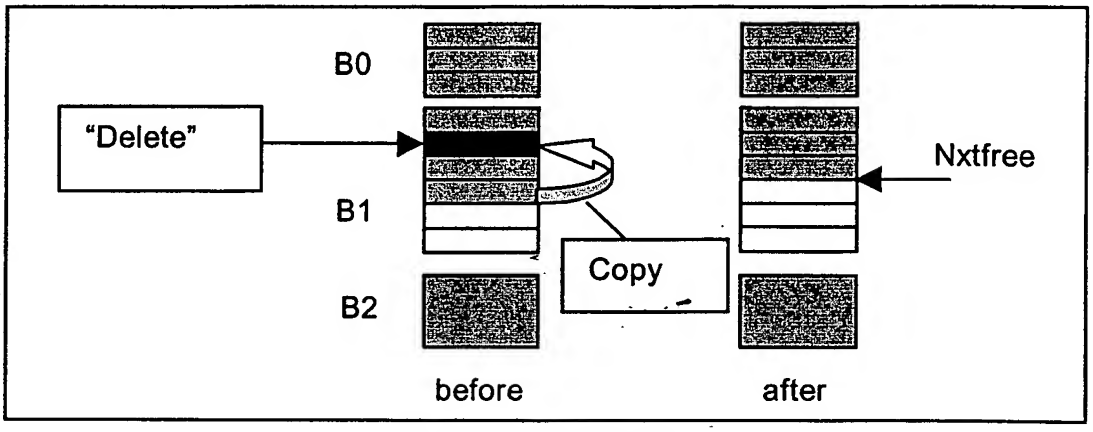

Figure 6: Deleting an Entry

match. When searching for the longest match, the Comparand register contents are duplicated in a Mask register and the search is performed with the relevant bits masked out.

When searching for explicit entries there is no need to use the Mask register. The ((~addr) & mask), (addr & mask) representation of the IP address/mask pair is loaded into the Comparand register and a compare is initiated. The address/mask pair is compared with the entries in the RCP address database. This search will only yield a match if the RCP word exactly matches the entry in the Comparand register. To search for an explicit RCP entry, the (addr&mask) is written into the lower part (bits 31–0) of the Comparand register (CR) and the ((~addr)&mask) is written into the higher part (bits 63–32) of the CR. Then a compare-operation is executed without the use of the Mask register (MR). The commands are :

```
WRL     CR {MR000}
CMPWH   DQ {MR000}
```

(Code descriptions can be found in Figure 10.)

Sorting Upwards

A special case to consider is when an entry is to be added to a full block and the blocks below are also full. When this occurs, entries cannot be swept further "downwards" to produce an empty location in the required block. Instead, they must be pushed up if there is any free space above.

A "backpressure" flag, mentioned earlier, is set in the last block when there is no more space available. The meaning of the "backpressure" flag is this: if it is not set, there is at least one more free entry somewhere between the current block and the highest physical memory location. If it is set, then all blocks below the current block are full. Blocks attempting to push downward check this flag and, if it is set, are forced to try to push entries upward. In this case, recursion downward into the blocks would not work because there is no more free memory. Instead, recursion must go upwards. The same sorting routine is applied here with the difference being that the sorting goes in the opposite direction. Figure 7 shows an example of recursive sorting upwards.

In A, a new entry is destined for the last block but it is full, requiring the edge sorting to go upward instead of downward. Secondly, B shows the moves and inserts that will be done to free an entry space in the final block. As before, the Routing database stored in the RCP will always yield a correct search result due to the order of the moves and inserts. Finally, C shows the blocks as represented by their readjusted pointers and block size values.

Interleaving Operations

It is possible to interleave operations of the RCP in order to keep the table updated while continuing to perform high bandwidth searches. The insertion operation can be broken up into different atomic operations that can be interleaved

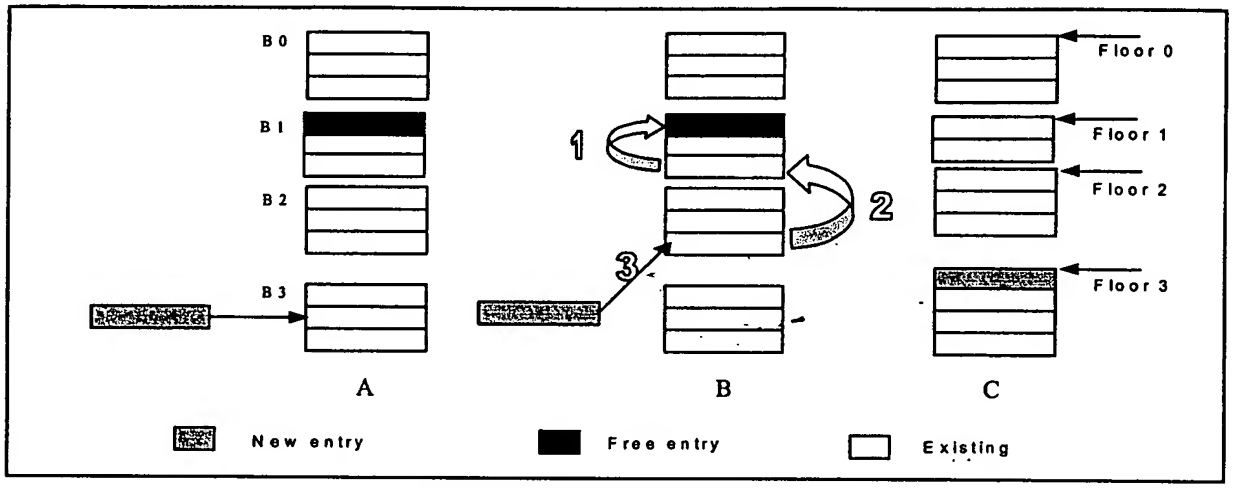

Figure 7: Recursive Sorting Upwards

in between gaps in the atomic search operations. The following enumerate the different atomic insert operations:

1. Insert to address:
   Two short cycles: insert (addr & mask) and ((~addr) & mask) to address

2. Read out entry to be copied/moved

Figure 8 shows how the four inserts and three copies required in the previous example on recursive sorting could be queued and interleaved in between address searches.

The worst case number of cycles for this can be calculated assuming that the new entry is inserted to the first block but all blocks except the last are full. For k partitions, there are:

k-1 MUAC read cycles
k MUAC write cycles
k RAM read/write cycles

An important point to note is that since k entries are moved, the overhead of moving their associated data in RAM must also be considered. However, this worst case only happens once; with thoughtful initial block allocation, inserting a new entry into the MUAC will take one write cycle. A good precautionary measure to avoid long sorting operations is to adjust the block size and spacing during idle periods to avoid shuffling during busy periods.

If the associated data entries stored in RAM are more than a few bytes in length, a scheme using a second section of RAM to incorporate pointers might be considered. The pointers would point to the associated data entries of each MUAC word entry, which would avoid moving large blocks of data. When the associated data entries are shuffled, it is a simple case of rearranging the pointers. This would remove the need to move the associated data around, which will reduce the associated data maintenance time.

ROUTING TABLE UPDATES

The method described in this Application Note is flexible enough to be used with different styles of updating the routing tables and therefore allows a system designer or software engineer the ability to use the most suitable method available. Using the algorithm described in this Application Note in conjunction with dynamic routing table updates would be more efficient than reloading the routing table in the MUAC and refilling it. This is because search operations could still be performed during the updates.

STORING IP-MULTICAST AND MAC ADDRESSES

In order to store the combination of addresses, each address "family" needs to have an identifier. It is unlikely that an IP address will start with a "don't care". This would mean the

other 31 bits would be "don't care" too, which addresses all IP-addresses available. So the bits 63 and 31 are never 00 nor 11. This combination can be used for identifying other address families. One should note that other assumptions could be made to "free" more bits, allowing even more possible identification fields. Examples of this are:

- The first 8 bits are almost never "don't care"
- The last 8 bits in IPv4 CIDR are almost always "don't care"

Such assumptions would of course lead to more flexibility; this is not detailed in this Application Note.

IP-Multicast

In this case a lookup on the combination of IP multicast destination address and IP source address are required. It It is suggested is to store the source address in the lower 32 bits of the RCP word (31–0) and the IP multicast destination address in the upper 32 (63–32). It is not necessary to store the entire 32-bit multicast destination address in the RCP, since Class D (=multicast) addresses always start with 1110. The IP source address can start with a 0 or with a 1 and since the ternary representation of the IPv4 (CIDR) addresses will always have a 01 or 10 in bits 63 and 31, bit 63 has to be set equal to bit 31 of the IP address pair to guarantee it does not look like a ternary value. For IP Multicast search operations bit 63 would have to be masked out. So all 32 bits of the IP source address are stored in bits 31–0 of a RCP word. From the IP multicast address bit 63 is being set to bit 31 of the source address. Bit 62–60 will always be 110. The routines for IP Multicast lookups follow. The structure of the addresses is shown in Figure 9.

Lookup of an IP Multicast Entry:

```
WRL     CR {MR000}
CMPWH   DQ {MR000}
```

(Code descriptions can be found in Figure 10.)

Insertion of an IP Multicast Entry-Routine:

```
WRL     [aaa]
WRH     [aaa]
```

(Code descriptions can be found in Figure 10.)

Deletion of an IP Multicast Entry-Routine:

```
WRL     CR {MR000}
CMPWH   DQ {MR000}
RST     V@[HPM]
```

(Code descriptions can be found in Figure 10.)

MAC Addresses

In bits 63 and 31, 00 is being stored also to guarantee that this entry does not look like a ternary value and therefore there is a difference between MAC addresses and the ternary IPv4 (CIDR) addresses. The difference between IP MultiCast addresses and MAC addresses is created by setting bit 62 of the MAC entry to 0. Bits 30–0 of the MAC address are stored into bits 30–0 of the RCP word entry, bits 47–32 of the MAC address into bits 47–32 of the RCP word entry and the remaining bit 31 of the MAC address into bit 48 of the RCP word. The bits 61–49 of the RCP word can be used to store additional information required for the layer 2 lookup. This additional information could be used for other purposes, such as a timestamp. The routines for MAC address lookup, updating, learning, and purging are written below. The structure in the RCP address database of the three address families is shown in Figure 9.

MAC Routines

A mask is required to mask out the bits 61–49 during the lookup (for example, MR005). Write the current time to the time stamp bits (for example, a TS of 8 bits into RCP word bits 61–54) so the TS of the MAC address (including the TS) can be updated when found or the MAC address (including the TS) can be learned (inserted). Another mask register (for example, MR006) needs to be programmed such that it masks of all bits except for the TS-bits (bits 61–54) and the bits 63 and 62, which must be 00 to concentrate the compare on the MAC-block in the RCP.

Note: Bits 63, 62, and 31 have to be 0 when performing MAC lookups; see Figure 9 for the complete diagram.

```
MR005 =   0x3FFE.0000
MR006 =   0xC03F.FFFF
```

MAC Address Lookup and Update/Learn Routine:

```
WRL     CR {MR000}
CMPWH   DQ {MR005}

;if match then update
MOV     [HPM],CR   {MR000}

;if no match then learn
WR      AR {MR000}
MOV     [AR],CR    {MR000}
```

(Code descriptions can be found in Figure 10.)

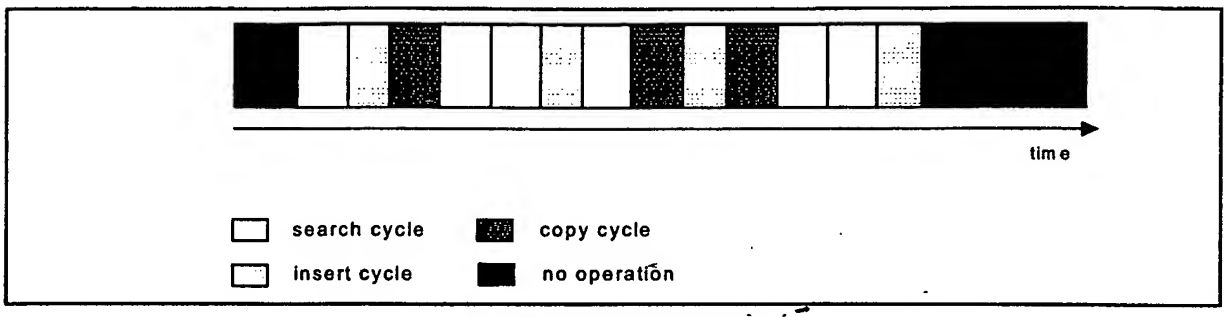

Figure 8: Cycle Interleaving

MAC Address Deletion Routine:
WRL CR {MR000}
CMPWH DQ {MR005}
RST V@[HPM]
(Code descriptions can be found in Figure 10.)

Purge Routine (Delete Aged (=old) MAC Addresses):
CMPWH DQ {MR006}
repeat until /MF is HIGH
RST V@AML
next
end repeat;
(Code descriptions can be found in Figure 10.)

INITIALIZATION ROUTINE

We assume that two 4K MUAC RCPs need to be cascaded. After a HW-reset the MUAC RCP is in SW-mode. First, the MUAC RCPs are set to HW-mode before the initialization can continue.

```
WR      FR {MR000}   (/AV=HIGH,/DQ12=LOW)
0x0000 0000          (/AV=LOW)
RST
WR      PA {MR000}   0x0000 0000
WR      PA {MR000}   0x0000 0001
RST     FF
WR      DS {MR000}   0x0000 0000
WR      FR {MR000}   0x0200 0000
WR      DS {MR000}   0x0000 0001
WR      FR {MR000}   0x0000 0001
```

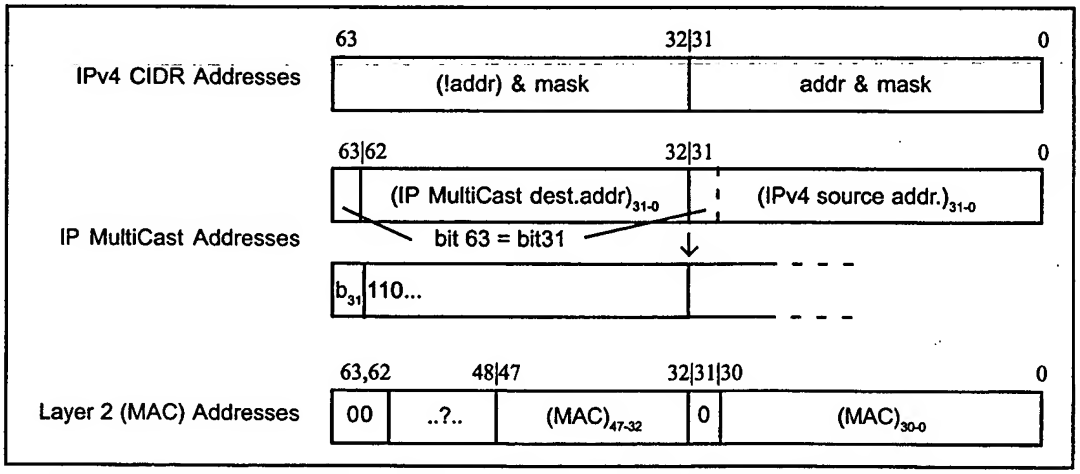

Figure 9: Structure of the Three Address Families When Stored in the RCP Address Database

The following two instructions are to initialize two mask registers, to be used when storing MAC addresses.

```
WR      MR005   0x3FFE 0000
WR      MR006   0xC03F FFFF
```

(Code descriptions can be found in Figure 10.)

The next steps depend on the data types that need to be stored. For example, data to be stored:

- IPv4 addresses with CIDR-masks ranging from /12 up to /28 (17 classes)
- IP multicast addresses
- MAC addresses

This tells us that the MUAC RCP needs to be divided into 19 blocks. The sizes of the IP Multicast- and MAC blocks can be predefined.

CONCLUSION

The MUAC RCP provides a high-speed and flexible Multi Layer alternative to software routing tables because of the ability to encode ternary values in a binary word and the ability to carry layer 2, layer 3, and layer 4 data in the list. By incorporating IPv4 CIDR in the manner described in this Application Note, table sizes can be greatly reduced and hence reduce component costs as compared with a more conventional IPv4 table. The possibility to store other addresses than IPv4 (CIDR) makes the MUAC RCP even more flexible and more powerful.

Insert a New IPv4 CIDR Address (with Mask):

```
WRL     [aaa]     ;the data on the DQ-bus (ip_addr_L) is written to the lower 32 bits of a
                  free address in the block
WRH     [aaa]     ;the data on the DQ-bus (ip_addr_H) is written to the higher 32 bits of a
                  free address in the block
```

CAM Operation Codes:

1. Two short cycles to move the entry that has to be moved from the original F1 location into a temporary register (CR).

```
WR      AR   {MR000}   ;write the location of the entry to be moved (AR=F1)
MOV     CR, [AR]       ;move the contents of this location into the CR
```

2. Two short cycles to move this entry into the next free location in the block.

```
WR      AR   {MR000}   ;write the location of the next free location in the block (AR=F1)
MOV     [AR],CR        ;move the contents of CR (ex F1) to this free location
```

3. Two short cycles to write the new entry into the newly created space.

```
WRL     [aaa]          ;write the lower 32 bits of the new entry into the newly created space
                       (aaa"=old" F1)
WRH     [aaa]          ;write the higher 32 bits of the new entry into the newly created space
```

Delete an IPv4 Address from the CAM:

```
WRL     CR   {MR000}   ;write the lower 32 bits (ip_addr_L) from the DQ-bus to the CR
CMPWH   DQ   {MR000}   ;write the higher 32 bits (ip_addr_H) from the DQ-bus to the CR and
                       starts the compare, the address of the matching entry has to be read
                       from the PA:AA-bus in order to update the list with free addresses of
                       this block
RST     V@[HPM]        ;deletes the matching entry
```

Searching for Explicit CAM Entry Command Codes:

```
WRL     CR   {MR000}   ;writes ip_src to the lower 32 bits of the CR
CMPWH   DQ   {MR000}   ;writes the ip_mc to the higher 32 bits and compares
```

Lookup of an IP Multicast Entry:

```
WRL     CR   {MR000}   ;writes ip_src to the lower 32 bits of the CR
CMPWH   DQ   {MR000}   ;writes the ip_mc to the higher 32 bits and compares
```

Insertion of an IP Multicast Entry-Routine:

```
WRL     [aaa]          ;writes the data on the DQ bus (ip_src) directly to the lower 32
                       bits of the location addressed by AC bus (a free location in the
                       Multicast block).
WRH     [aaa]          ;writes the data on the DQ bus (ip_mc) directly to the higher 32 bits of
                       the location addressed by AC bus (a free location in the Multicast
                       block); mind bits 63, 62, 61, and 60.
```

Figure 10: RCP Description Codes

```
Deletion of an IP Multicast Entry-Routine:
   WRL          CR      {MR000}    ;writes ip_src to the lower 32 bits of the CR
   CMPWH        DQ      {MR000}    ;writes the ip_mc to the higher 32 bits and compares; the address of
                                    the matching entry has to be read from the PA:AA-bus in order to
                                    update the list with free addresses of this block
   RST          V@[HPM]            ;resets the validity bit of the matching location

MAC Address Lookup and Update/Learn Routine:
   WRL          CR      {MR000}    ;writes mac_addr_L into the CR
   CMPWH        DQ      {MR005}    ;writes mac_addr_H into the CR and launches the compare
                                    through MR005.
;if match then update
   MOV          [HPM],CR{MR000}    ;if the address was found the TS needs to be updated, so the contents
                                    of the CR are moved to the matching location; the address of this location
                                    is available on the PA:AA bus
;if no match then learn
   WR           AR      {MR000}    ;write the address of a free location in the MAC-block to the
                                    address register
   MOV          [AR],CR {MR000}    ;if the address was NOT found is should be learned by moving the
                                    contents to a free location in the MAC-block of the RCP, the address
                                    of this location is available on the PA:AA bus

MAC Address Deletion Routine:
   WRL          CR      {MR000}    ;write the 32 lower bits into the CR
   CMPWH        DQ      {MR005}    ;write the 32 higher bits into the CR and launches the compare
                                    through MR005, mind that bits 63,62, and 31 have to be 0.
   RST          V@[HPM]            ;reset the validity bit of the matching location

Purge Routine (Delete Aged (=Old) MAC Addresses):
   CMPWH        DQ      {MR006}    ;write 00 and TS to the higher part of CR and launch a compare
                                    through MR006
   ;repeat until /MF is HIGH
   RST          V@HMA              ;resets the validity bit of the matching location, the address of this
                                    location has to read from the PA:AA bus to keep track of the free
                                    entries in this block
   NEXT;                           proceed to the next matching entry
   ;end repeat
```

Figure 10: RCP Description Codes *Continued*

```
Initialization Routine:
   WR      FR      {MR000}    (/AV=HIGH, /DQ12= LOW)
   0x0000 0000                (/AV=LOW)        ;sets the MUAC RCP to HW-Mode

   RST                                         ;SW-reset
   WR      PA      {MR000}    0x0000 0000      ;sets the PA of the highest priority device to "0";
                                               and sets /FF LOW
   WR      PA      {MR000}    0x0000 0001      ;sets the PA of the highest priority device to "1";
                                               and sets /FF LOW
   RST     FF;                                 resets all Full Flags (/FF HIGH)
   WR      DS      {MR000}    0x0000 0000      ;select RCP0
   WR      FR      {MR000}    0x0200 0000      ;sets RCP0 to use no def.mask and sets it to
                                               NOT LOW priority RCP
   WR      DS      {MR000}    0x0000 0001      ;selects RCP1
   WR      FR      {MR000}    0x0000 0001      ;sets RCP0 to use no def.mask and sets it to
                                               LOW priority RCP

   WR      MR005   0x3FFE 0000                 ;sets MR005 to mask out bits 61–49
   WR      MR006   0xC03F FFFF                 ;sets MR006 to mask out all except bits 61–49
```

Figure 10: RCP Description Codes *Continued*

MUSIC Semiconductors Agent or Distributor:

MUSIC SEMICONDUCTORS
http://www.music-ic.com
email: info@music-ic.com

USA Headquarters
MUSIC Semiconductors
254 B Mountain Avenue
Hackettstown, New Jersey 07840
USA
Tel: 908/979-1010
Fax: 908/979-1035
USA Only: 800/933-1550 Tech. Support
888/226-6874 Product Info.

Asian Headquarters
MUSIC Semiconductors
Special Export Processing Zone 1
Carmelray Industrial Park
Canlubang, Calamba, Laguna
Philippines
Tel: +63 49 549 1480
Fax: +63 49 549 1023
Sales Tel/Fax: +632 723 62 15

European Headquarters
MUSIC Semiconductors
Torenstraat 28
6471 JX Eygelshoven
Netherlands
Tel: +31 45 5462177
Fax: +31 45 5463663

Docket No.: M4065.0573/P573-B
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
David C. Feldmeier et al.

Application No.: 10/645,604

Group Art Unit: 2186

Filed: August 22, 2003

Examiner: Matthew D. Anderson

For: PARTIALLY-ORDERED CAMS USED IN TERNARY HIERARCHICAL ADDRESS SEARCHING/SORTING

DECARATION UNDER 37 C.F.R. § 1.132

Commissioner for Patents
Washington, DC 20231

Dear Sir:

I, David C. Feldmeier, of Redwood City, CA, declare as follows:

that I was an employee of Music Semiconductors, Inc., formerly of Hackettstown NJ, San Jose CA, and Milpitas CA (Music Semiconductors) during the time interval from July 1996 to March 2002;

that as an employee of Music Semiconductors I made inventions, and prepared documents for publication relating to such inventions;

that I am an Applicant and Co-inventor (with Tyler Arnold) of United States Patent Application No. 10/645,604 (the Application) entitled Partially-Ordered Cams Used in Ternary Hierarchical Address Searching/Sorting;

that, as Co-inventors, Tyler Arnold and I assigned the Application to Music Semiconductors;

that, on information and belief, the Application has now been assigned to Micron Technology, Inc., of Boise Idaho;

that I have reviewed now-pending claims 56-87 of the Application, a copy of which is attached hereto, and affirm that Tyler Arnold and I are Co-inventors of the invention claimed therein;

that I am the Co-author (with Tyler Arnold) of "A Method* For Fast IPv4 CIDR Address Translation and Filtering Using the MUSIC WidePort LANCAM, LANCAM, and LANCAM 1st Family," dated January 1998 (the LANCAM Publication), a copy of which is attached hereto;

that I am also the Co-author (with Tyler Arnold) of "Fast IPv4 And IPv4 CIDR Address Translation And Filtering Using the MUAC™ Routing Coprocessor (RCP)*," dated January 1998 (the MUAC Publication), a copy of which is attached hereto;

that Co-authors of the LANCAM Publication and MUAC Publication, Tyler Arnold and I, disclosed therein the invention as now claimed in claims 56-87 of the Application; and accordingly

that (1) the subject matter of the LANCAM Publication relevant to the claims (Relevant Subject Matter) was derived from the Applicants, (2) the subject matter of the MUAC Publication relevant to the claims (Relevant Subject Matter) was derived from the Applicants and (3) the Applicants actually invented the Relevant Subject Matter;

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; further, that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment or both, under section 1001 of Title 19 of the U.S. Code and that such willful statement may jeopardize the validity of the application or any patent issued thereon.

Further declarant sayeth not.

David C. Feldmeier

Date: February 8, 2005

LISTING OF THE CLAIMS

56. A binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said binary CAM comprising a plurality of entries wherein:

each entry comprises:

(1) a first value comprising the logically ANDed communication system address and its associated mask; and

(2) a second value comprising the logically ANDed complement of said communication system address and its associated mask; and

wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

57. The binary CAM of claim 56 wherein entries having the most amount of contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

58. The binary CAM of claim 57 wherein said first value comprises n bits and said second value comprises n bits and wherein each one of said bits in said first value has an associated bit in said second value, said each one of said bits and said associated bit

forming a binary pair, said binary pair representing one bit of said address as two bits in said CAM.

59. The binary CAM of claim 58 wherein a 1 in said ternary hierarchical address is represented as 10 in said CAM, wherein a 0 in said ternary hierarchical address is represented as 01 in said CAM, and wherein a don't care in said ternary hierarchical address is represented as 00 in said CAM.

60. The binary CAM of claim 58 wherein said first value is stored in an upper portion of said entry and said second value is stored in a lower portion of said entry.

61. The binary CAM of claim 60 wherein said entry comprises 64 bits and wherein n is 32.

62. A binary content addressable memory (CAM) for storing ternary hierarchical addresses of a communication system therein and wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said binary CAM comprising a plurality of entries wherein:

each entry comprises a two bit representation for each bit in said ternary hierarchical address; and

wherein each entry is positioned in said CAM based on the number of contiguous ones in said associated mask.

63. The binary CAM of claim 62 wherein:

(a) a 1 in said ternary hierarchical address is represented by said two bit representation in said CAM as 10;

(b) a 0 in said ternary hierarchical address is represented by said two bit representation in said CAM as 01; and

(c) a don't care in said ternary hierarchical address in represented by said two bit representation in said CAM as 00.

64. The binary CAM of claim 63 wherein entries having the most amount of contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

65. A method for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said method comprising the steps of:

logically ANDing each communication system address and its associated mask to form a first value;

logically ANDing the complement of each communication system address and its associated address mask to form a second value; and

storing said first value and said second value in an entry in said binary CAM based on the number of contiguous ones in said address mask.

66. The method of claim 65 wherein said step of storing said first value and said second value at a location comprises arranging entries such that entries having the most contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

67. The method of claim 66 wherein said first value comprises n bits and said second value comprises n bits and wherein said step of storing said first value and said second value comprises associating each one of said bits in said first value with one bit in said second value to form a binary pair, said binary pair representing one bit of said address as two bits in said CAM.

68. The method of claim 67 wherein said step of representing one bit of said ternary hierarchical address as two bits in said CAM comprises:

(a) utilizing 10 in said CAM to represent a 1 in said ternary hierarchical address;

(b) utilizing 01 in said CAM to represent a 0 in said ternary hierarchical address; and

(c) utilizing 00 in said CAM to represent a don't care in said ternary hierarchical address.

69. The method of claim 67 wherein said step of storing said first value and said second value comprises storing said first value in an upper portion of said entry and storing said second value in a lower portion of said entry.

70. The method of claim 69 wherein said entry comprises 64 bits and wherein said step of storing said first value and said second value comprises storing said first value in the upper 32 bits and storing said second value in the lower 32 bits.

71. A method of storing for storing ternary hierarchical addresses of a communication system in a binary CAM wherein each ternary hierarchical address comprises a communication system address and associated communication system address mask, said method comprising the steps of:

(a) generating a CAM entry for each ternary hierarchical address by:

(1) identifying that portion of each ternary hierarchical address that are don't cares;

(2) representing each bit in said ternary hierarchical address using bits in said CAM wherein:

(i) a 1 in said ternary hierarchical address is represented as 10 in said CAM;

(ii) a 0 in said ternary hierarchical address is represented as 01 in said CAM; and

(iii) a don't care in said ternary hierarchical address is represented as .00 in said CAM;

(b) positioning each CAM entry in said CAM based on the number of contiguous ones in said associated mask.

72. The method of claim 71 wherein said step of positioning each CAM entry in said CAM comprises arranging entries such that entries having the most contiguous ones in said associated mask are located at the top of said CAM and wherein entries having the least amount of contiguous ones in said associated mask are located at the bottom of said CAM.

73. A method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address entry comprising a communication system address and an associated communication system address mask, said entries of ternary hierarchical addresses being stored in said binary CAM as a first value and a second value, said first value comprising the communication system address logically ANDed with said address mask and said second value comprising the complement of said communication system address logically ANDed with said address

mask, and wherein all of said entries are ordered in said binary CAM based on the number of contiguous ones in said mask, said method comprising the steps of:

(a) loading a first register with the communication system address to be searched along with the complement of the communication system address;

(b) loading a second register with the communication system address to be searched along with the complement of the communication system address;

(c) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM;

(d) determining whether a bit match occurs between corresponding bits of said first register and each entry in said binary CAM as qualified by said corresponding bit of said second register; and

(e) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

74. The method of claim 73 wherein said step of determining whether a bit match occurs between corresponding bits of said first register and each entry in said binary CAM as qualified by said corresponding bit of said second register comprises declaring a bit match between corresponding bits of said first register and each entry in said binary CAM:

(1) if the corresponding bit in said second register is a 1; or

(2) if the corresponding bit in said second register is a 0 and the corresponding bits of said first register and each entry in said binary CAM are identical.

75. The method of claim 74 wherein said steps of loading a first and second register comprises loading the communication system address to be searched in an upper portion of said registers and loading said complement of the communication system address in a lower portion of said registers.

76. The method of claim 75 wherein said first and second registers comprise 64 bits and wherein said steps of loading said first and second registers comprises loading the communication system address to be searched in the upper 32 bits of said registers and loading said complement of the communication system address in said lower 32 bits of said registers.

77. A method of searching a binary CAM to find a match for a ternary hierarchical address of a communication system, said binary CAM comprising entries of ternary hierarchical addresses, each ternary hierarchical address comprising a communication system address and an associated communication system address mask, each ternary hierarchical address entry being stored in said binary CAM whereby 10 is stored in said entry for every 1 in said ternary hierarchical address, 01 is stored in said entry for every 0 in said ternary hierarchical address, and 00 is stored in said entry for every don't care in said ternary hierarchical address, and wherein all of said entries are ordered in said

binary CAM based on the number of contiguous ones in said mask, said method comprising the steps of:

(a) loading a first register and second register with the communication system address to be searched by:

(1) loading a 10 in said first and second registers for every 1 in the ternary hierarchical address to be searched;

(2) loading a 01 in said first and second registers for every 0 in the ternary hierarchical address to be searched;

(3) loading a 00 in said first and second registers for every don't care in the ternary hierarchical address to be searched;

(b) associating each bit of said first register with one bit of said second register and with one bit of each entry in said binary CAM;

(c) declaring a bit match between corresponding bits of said first register and each entry in said binary CAM:

(1) if the corresponding bit in said second register is a 1; or

(2) if the corresponding bit in said second register is a 0 and the corresponding bits of said first register and each entry in said binary CAM are identical; and

(d) obtaining a match between the desired ternary hierarchical address and one of said entries based on the greatest number of matches of corresponding bits of said first register and each entry in said binary CAM.

78. A method for maintaining a sorted CAM to enable longest matches in a single search cycle when hierarchical addresses are added to, or deleted from, the CAM in a communication system utilizing hierarchical addresses and associated address masks, said method comprising the steps of:

(a) segmenting the CAM into blocks wherein each block corresponds to a single hierarchical mask and wherein said blocks are arranged in the CAM such that the lowest CAM addresses contain the highest hierarchical masks and the highest CAM addresses contain the lowest hierarchical masks;

(b) storing hierarchical addresses according to said block having a corresponding hierarchical mask; and

(c) tracking the first address and the next free address of each of said blocks and the size of each of said blocks.

79. A content addressable memory (CAM) of a communications system utilizing ternary hierarchical addressing and associated address masks, said CAM comprising:

a plurality of address entries and associated address masks that are arranged in said CAM by mask number, with address entries having the highest mask number being located at address entry locations at the top of the CAM and address entries having the lowest mask number being located at address entry locations at the bottom of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

80. An apparatus for storing a plurality of address entries and associated address masks in a communication system utilizing ternary hierarchical addressing and associated address masks, said apparatus comprising:

a binary CAM;

a binary-encoded ternary converter coupled to said binary CAM wherein said binary-encoded ternary converter converts each ternary value into a corresponding binary-encoded ternary value to form said address entries; and

wherein said plurality of address entries and associated address masks are arranged in said binary CAM in a plurality of address entry groups, each address entry group having a respective mask number, with said address entry group having the highest mask number being located at the highest location of the CAM and said address entry group having the lowest mask number being located at the lowest location of the CAM, said mask number being defined as the number of contiguous ones in an associated address mask.

81. A method for accelerating the routing of hierarchical addressing in a communication system which utilizes ternary hierarchical addressing and associated address masks, said method comprising the steps of:

(a) obtaining communication system hierarchical addresses and associated masks to form a plurality of address entries; and

(b) storing said plurality of address entries in a content-addressable memory (CAM) device by mask number wherein said mask number is defined as the number of

contiguous ones in an associated address mask and wherein address entries having the highest mask number are stored in address entry locations at the highest location of the CAM and address entries having the lowest mask number are stored at address entry locations at the lowest location of the CAM.

82. A content addressable memory (CAM) of a communication system as defined in claim 79 wherein said plurality of address entries comprises:

a first group of address entries sharing a first said address mask and a second group of address entries sharing a second said address mask, said first and second groups being located at different locations of the CAM.

83. A content addressable memory (CAM) of a communication system as defined in claim 82 further comprising at least one vacant address entry location disposed within said CAM between said first and second groups of address entries.

84. An apparatus for storing a plurality of address entries and associated address masks in a communication system as defined in claim 80 further comprising:

at least one vacant address entry location within said CAM adjacent at least one of said address entry groups.

85. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 81 further comprising the steps of:

receiving an address value in a comparand register of said CAM;

matching a further plurality of addresses, including a subset of said communication system hierarchical addresses, to said address value; and

outputting an output hierarchical address of said further plurality of addresses according to respective storage locations of said further plurality of addresses.

86. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 85 wherein said output hierarchical address of said further plurality has a storage location lowest among said respective storage locations of said further plurality of addresses.

87. A method for accelerating the routing of hierarchical addressing in a communication system as defined in claim 85 wherein said further plurality of said communication system hierarchical addresses each includes at least one identical bit.

APPENDIX C – RELATED PROCEEDINGS

NONE